

LNCs

LNC

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

218

Advances in Cryptology – CRYPTO '85

Proceedings

Edited by Hugh C. Williams



Springer-Verlag
Berlin Heidelberg New York Tokyo

Editorial Board

D. Barstow W. Brauer P. Brinch Hansen D. Gries D. Luckham
C. Moler A. Pnueli G. Seegmüller J. Stoer N. Wirth

Editor

Hugh C. Williams
Department of Computer Science, University of Manitoba
Winnipeg, Manitoba R3T 2N2, Canada

CR Subject Classifications (1985): E.3

ISBN 3-540-16463-4 Springer-Verlag Berlin Heidelberg New York Tokyo

ISBN 0-387-16463-4 Springer-Verlag New York Heidelberg Berlin Tokyo

CIP-Kurztitelaufnahme der Deutschen Bibliothek: Advances in cryptology: proceedings of
CRYPTO ... – Berlin; Heidelberg; New York; Tokyo: Springer.
Teilw. in d. Vorlage auch: Workshop on the Theory and Application of Cryptograph. Techniques
NE: CRYPTO 1985 (1986).
(Lecture notes in computer science: Vol. 218)
ISBN 3-540-16463-4 (Berlin ...)
ISBN 0-387-16463-4 (New York ...)
NE: GT

This work is subject to copyright. All rights are reserved, whether the whole or part of the material
is concerned, specifically those of translation, reprinting, re-use of illustrations, broadcasting,
reproduction by photocopying machine or similar means, and storage in data banks. Under
§ 54 of the German Copyright Law where copies are made for other than private use, a fee is
payable to "Verwertungsgesellschaft Wort", Munich.

© by Springer-Verlag Berlin Heidelberg 1986
Printed in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210

Preface

In the summer of 1981 Allen Gersho organized the first major open conference ever devoted to cryptologic research. This meeting, Crypto '81, was held at the University of California campus in Santa Barbara. Since then the Crypto' conference has become an annual event. These are the proceedings of the fifth¹ of these conferences, Crypto '85.

Each section of this volume corresponds to a session at the meeting. The papers were accepted by the program committee, sometimes on the basis of an abstract only, and appear here without having been otherwise refereed. The last section contains papers for some of the impromptu talks given at the traditional rump session. Each of these papers was refereed by a single member of the program committee. An author index as well as a keyword index, the entries for which were mainly supplied by the authors, appear at the end of the volume.

Unfortunately, two of the papers accepted for presentation at Crypto '85 could not be included in this book they are:

Unique Extrapolation of Polynomial Recurrences

J.C. Lagarias and J.A. Reeds (A.T. & T Bell Labs)

Some Cryptographic Applications of Permutation Polynomials and
Permutation Functions

Rupert Nöbauer (Universität für Bildungswissenschaften, Austria)

It is my great pleasure to acknowledge the efforts of all of those who contributed to making these proceedings possible: the authors, program committee, other organizers of the meeting, IACR officers and directors, and all the attendees. I would also like to thank Lynn Montz of Springer-Verlag for her patient assistance in preparing this volume.

Winnipeg, Manitoba, Canada
January 1986

H.C.W.

¹Proceedings of the other Crypto conferences have also been published. The interested reader can find these listed in the preface of Advances in Cryptology 84 (the proceedings of Crypto '84), published by Springer-Verlag.

CRYPTO 85

A Conference on the Theory and Application of Cryptographic Techniques

held at the University of California, Santa Barbara,
through the co-operation of the
Computer Science Department

August 18-22, 1985

sponsored by

The International Association for Cryptologic Research

in co-operation with

*The IEEE Computer Society Technical Committee
on Security and Privacy*

Organizers

Ernest F. Brickell (Bell Communications Research), General Chairman
H.C. Williams (University of Manitoba), Program Chairman
Thomas A. Berson (Sytek, Inc.), Program
Joan Boyar (University of Chicago), Program
Donald W. Davies (Data Security Consultant), Program
Oded Goldreich (MIT/Technion), Program
Alan G. Konheim (UCSB), Local Arrangements
Carol Patterson (Sandia Laboratories), Registration
Ron Rivest (MIT), Program
Joe Tardo (DEC), Show and Tell

CONTENTS

SECTION I: SIGNATURES AND AUTHENTICATION

Breaking the Ong-Schnorr-Shamir Signature Scheme for Quadratic Number Fields	3
<i>Dennis Estes, Leonard M. Adleman, Kireeti Kompella, Kevin S. McCurley, and Gary L. Miller</i>	
Another Birthday Attack	14
<i>Don Coppersmith</i>	
Attacks on Some RSA Signatures	18
<i>Wiebren de Jonge and David Chaum</i>	
An Attack on a Signature Scheme Proposed by Okamoto and Shiraishi	28
<i>Ernest F. Brickell and John M. DeLaurentis</i>	
A Secure Subliminal Channel (?)	33
<i>Gustavus J. Simmons</i>	
Unconditionally Secure Authentication Schemes and Practical and Theoretical Consequences	42
<i>Yvo Desmedt</i>	

SECTION II: PROTOCOLS

On the Security of Ping-Pong Protocols When Implemented Using the RSA	58
<i>Shimon Even, Oded Goldreich, and Adi Shamir</i>	
A Secure Poker Protocol that Minimizes the Effect of Player Coalitions	73
<i>Claude Crepeau</i>	
A Framework for the Study of Cryptographic Protocols	87
<i>Richard Berger, Sampath Kannan, and Rene Peralta</i>	
Cheating at Mental Poker	104
<i>Don Coppersmith</i>	
Security for the DoD Transmission Control Protocol	108
<i>Whitfield Diffie</i>	
Symmetric Public-Key Encryption	128
<i>Zvi Galil, Stuart Haber, and Moti Yung</i>	

SECTION III: COPY PROTECTION

Software Protection: Myth or Reality?	140
<i>James R. Gosler</i>	
Public Protection of Software	158
<i>A. Herzberg and S. Pinter</i>	
Fingerprinting Long Forgiving Messages	180
<i>G.R. Blakley, Catherine Meadows, and G.B. Purdy</i>	

SECTION IV: SINGLE KEY CRYPTOLOGY

Cryptanalysis of DES with a Reduced Number of Rounds	192
<i>David Chaum and Jan-Hendrik Evertse</i>	
Is DES a Pure Cipher? (Results of More Cycling Experiments on DES)	212
<i>Burt S. Kaliski, Ronald L. Rivest, and Alan T. Sherman</i>	
A Layered Approach to the Design of Private Key Cryptosystems	227
<i>T.E. Moore and S.E. Tavares</i>	
Lifetimes of Keys in Cryptographic Key Management Systems	246
<i>E. Okamoto and K. Nakamura</i>	
Correlation Immunity and the Summation Generator	260
<i>Rainer A. Rueppel</i>	
Design of Combiners to Prevent Divide and Conquer Attacks	273
<i>T. Siegenthaler</i>	
On the Security of DES	280
<i>Adi Shamir</i>	
Information Theory Without the Finiteness Assumption, II Unfolding the DES	282
<i>G.R. Blakley</i>	

SECTION V: TWO KEY CRYPTOLOGY

Analysis of a Public Key Approach Based on Polynomial Substitution	340
<i>Harriet Fell and Whitfield Diffie</i>	
Developing an RSA Chip	350
<i>Martin Kochanski</i>	

An M^3 Public-Key Encryption Scheme	358
<i>H.C. Williams</i>	
Trapdoor Rings and Their Use in Cryptography	369
<i>V. Varadharajan</i>	
On Computing Logarithms Over Finite Fields	396
<i>Taher El Gamal</i>	
On Using RSA with Low Exponent in a Public Key Network	403
<i>Johan Hastad</i>	
Lenstra's Factorisation Method Based on Elliptic Curves	409
<i>N.M. Stephens</i>	
Use of Elliptic Curves in Cryptography	417
<i>Victor S. Miller</i>	

SECTION VI: RANDOMNESS AND OTHER PROBLEMS

Cryptography with Cellular Automata	429
<i>Stephen Wolfram</i>	
Efficient Parallel Pseudo-Random Number Generation	433
<i>J.H. Reif and J.D. Tygar</i>	
How to Construct Pseudo-random Permutations from Pseudo-random Functions	447
<i>Michael Luby and Charles Rackoff</i>	
The Bit Security of Modular Squaring Given Partial Factorization of the Modulus	448
<i>Benny Chor, Oded Goldreich, and Shafi Goldwasser</i>	
Some Cryptographic Aspects of Womcodes	458
<i>Philippe Godlewski and Gerard D. Cohen</i>	
How to Reduce Your Enemy's Information	468
<i>Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert</i>	
Encrypting Problem Instances: Or ... Can you Take Advantage of Someone Without Having to Trust Him?	477
<i>Joan Feigenbaum</i>	
Divergence Bounds on Key Equivocation and Error Probability in Cryptanalysis	489
<i>J. van Tilburg and D.E. Boeke</i>	

SECTION VII: IMPROMPTU TALKS

A Chosen Text Attack on the RSA Cryptosystem and Some Discrete Logarithm Schemes	516
<i>Y. Desmedt and A.M. Odlyzko</i>	
On the Design of S-boxes	523
<i>A.F. Webster and S.E. Tavares</i>	
The Real Reason for Rivest's Phenomenon	535
<i>Don Coppersmith</i>	
The Importance of "Good" Key Scheduling Schemes (How to Make a Secure DES Scheme with ≤ 48 Bit Keys?)	537
<i>J.-J. Quisquater, Y. Desmedt, and M. Davio</i>	
Access Control at the Netherlands Postal and Telecommunications Services	543
<i>W. Haemers</i>	
Author Index	545
Keyword Index	546

Breaking the Ong-Schnorr-Shamir Signature Scheme for Quadratic Number Fields

Dennis Estes ⁽¹⁾
Leonard M. Adleman ^{(2)(*)}
Kireeti Kompella ⁽²⁾
Kevin S. McCurley ⁽¹⁾
Gary L. Miller ⁽²⁾

⁽¹⁾ Department of Mathematics
University of Southern California
Los Angeles, CA 90089-1113

⁽²⁾ Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0782

1. Introduction

Recently Ong, Schnorr, and Shamir [OSS1, OSS2] have presented new public key signature schemes based on quadratic equations. We will refer to these as the OSS schemes. The security of the schemes rest in part on the difficulty of finding solutions to

$$x^2 - KY^2 \equiv M \pmod{n}, \quad (1)$$

where n is the product of two large rational primes. In the original OSS scheme [OSS1], K , M , X , and Y were to be rational integers. However, when this version succumbed to an attack by Pollard [PS,S1], a new version was introduced [OSS2], where M , X , and Y were to be quadratic integers, i. e. elements of the ring $\mathbb{Z}[\sqrt{d}]$. In this paper we will show that the OSS system in $\mathbb{Z}[\sqrt{d}]$ is also breakable. The method by which we do this is to

*Research sponsored by NSF Grant *53-4510-2651

reduce the problem of solving the congruence over the ring $\mathbb{Z}[\sqrt{d}]$ to the problem of solving the congruence over the integers, for which we can use Pollard's algorithm.

The OSS signature scheme described in [OSS2] was intended to provide a method by which a person can sign messages with the assurance that no one, including the receiver, can forge the signature, and so that anyone can easily verify the validity of both the signature and the message. It works as follows: Party A generates two rational primes p and q each about 300 bits long, using the same care as in the key generation for RSA to ensure that $n = pq$ cannot be easily factored by known methods. Party A also chooses random integers d , t_0 , and t_1 such that $(n, t_0^2 - dt_1^2) = 1$, and publishes n , d , and $K \equiv (t_0 + t_1\sqrt{d})^2 \pmod{n}$, keeping t_0 , t_1 , p , and q secret. (In [OSS2], they took $K \in \mathbb{Z}$, but we will show that the scheme is insecure with $K \in \mathbb{Z}[\sqrt{d}]$). The messages consist of pairs of integers (M_0, M_1) from the interval $[1, n)$. In order to sign a message, party A uses the secret key (t_0, t_1) to construct a solution to the congruence $x^2 - Ky^2 \equiv M_0 + M_1\sqrt{d} \pmod{n}$. The receiver of the message can easily verify that the message was the one signed by party A. In order for the scheme to be secure, the receiver should have some assurance that no one can forge the signature without knowledge of the secret key K .

It was presumed that it would be hard to solve congruence (1) without knowing the secret keys, in part because $\mathbb{Z}[\sqrt{d}]$ is not in general a Euclidean domain, and Pollard's algorithm resembles the Euclidean algorithm in some ways. In this paper we will show that the problem of solving (1) over $\mathbb{Z}[\sqrt{d}]$ can be reduced to the problem of solving (1) over \mathbb{Z} . Pollard's algorithm can then be used to solve the problem over \mathbb{Z} , giving then also a solution over $\mathbb{Z}[\sqrt{d}]$. Because we use Pollard's algorithm, the method constructs a solution to the congruence without necessarily producing the secret keys. The most general OSS scheme was based on a polynomial congruence modulo a composite integer. Even though both of the quadratic OSS schemes have now been broken, it remains an open question whether the most general form of the OSS scheme can be broken.

In this paper, when we write $x_0 + x_1\sqrt{d} \equiv y_0 + y_1\sqrt{d} \pmod{n}$ we mean that $x_0 \equiv y_0 \pmod{n}$ and $x_1 \equiv y_1 \pmod{n}$. In general, an element X of the ring $\mathbb{Z}[\sqrt{d}]$ will be written as $X = x_0 + x_1\sqrt{d}$, and we will write $N(X)$ for the norm of X , namely

$N(X) = X_0^2 - dX_1^2$. If $X \in \mathbb{Z}[\sqrt{d}]$ and $(N(X), n) = 1$, then X is invertible modulo n , and we write $\bar{X} = \bar{X}_0 + \bar{X}_1\sqrt{d}$ for the inverse. Note that

$$\bar{X}_0 \equiv X_0 N(X)^{-1} \pmod{n},$$

$$\bar{X}_1 \equiv -X_1 N(X)^{-1} \pmod{n},$$

and these can be calculated using the Euclidean algorithm, even though $\mathbb{Z}[\sqrt{d}]$ may not be a Euclidean domain.

To begin, we consider four computational problems:

Problem I

INPUT $A, M, n \in \mathbb{Z}$, $(n, A) = (n, M) = 1$.

OUTPUT $X, Y \in \mathbb{Z}$ such that $X^2 + AY^2 \equiv M \pmod{n}$.

Problem II

INPUT $A, B, C, M, n \in \mathbb{Z}$, $n \nmid A$, $n \nmid B$, $n \nmid (C^2 - AB)$, $n \nmid M$.

OUTPUT Either a) or b):

a) $X, Y \in \mathbb{Z}$ such that $AX^2 + BY^2 + 2CXY \equiv M \pmod{n}$.

b) $m \in \mathbb{Z}$ such that $1 < m < n$ and $m \mid n$.

Problem III

INPUT $d, n \in \mathbb{Z}$, $K = K_0 + K_1\sqrt{d}$, $M = M_0 + M_1\sqrt{d}$, $(N(KM), n) = 1$, $n \nmid M_1$, $n \nmid d$.

OUTPUT Either a), b), or c):

a) $X, Y \in \mathbb{Z}[\sqrt{d}]$, $c \in \mathbb{Z}$ such that $(c, n) = 1$ and $X^2 - KY^2 \equiv cM \pmod{n}$.

b) $m \in \mathbb{Z}$ such that $1 < m < n$ and $m \mid n$.

c) $S \in \mathbb{Z}[\sqrt{d}]$ such that $S^2 \equiv K \pmod{n}$.

Problem IV

INPUT: $d, n \in \mathbb{Z}$, $K, M \in \mathbb{Z}[\sqrt{d}]$, $(N(KM), n) = 1$.

OUTPUT: $X, Y \in \mathbb{Z}[\sqrt{d}]$ with $X^2 - KY^2 \equiv M \pmod{n}$.

We shall prove:

Theorem 1 Problem II is solvable in polynomial time with an oracle for Problem I.

Theorem 2 Problem III is solvable in polynomial time with an oracle for Problem II.

Theorem 3 Problem IV is solvable in polynomial time with an oracle for Problem III.

The security of the original OSS scheme was based on the difficulty of solving Problem I when $N = pq$, the product of two large primes. Pollard produced an algorithm for Problem I which is believed to run in deterministic polynomial time, and as a consequence was able to break the original OSS signature scheme. The details of his algorithm should appear in a joint paper of Pollard and Schnorr [PS], where they will prove under the assumption of an extended Riemann hypothesis that Pollard's algorithm for Problem I runs in random polynomial time. It should be mentioned that in this paper they also outline a method similar to ours for solving Problem III, having made this discovery independently of the authors.

Three of the authors (A., E. and Mc.) have recently discovered a variation of Pollard's algorithm that allows us to prove that Problem I is solvable in random polynomial time, removing the assumption of the extended Riemann hypothesis in Pollard and Schnorr's result. Our variation of Pollard's algorithm is not a practical procedure for breaking the OSS scheme, but it has the advantage that one can rigorously analyse its running time without any hypothesis. The details of this will appear in a later paper.

As a consequence of these results, it follows that Problem IV is solvable in random polynomial time, and therefore that the OSS signature scheme over $\mathbb{Z}[\sqrt{d}]$ is insecure.

Several remarks are in order here before we proceed.

1. The assumption that $(N(KM), n) = 1$ is made primarily for convenience. In the OSS signature scheme, n was taken to be the product of two large primes, and the scheme is compromised if the factorization of n can be discovered. Therefore values of $N(K)$ and $N(M)$ which have a nontrivial factor in common with n are not

of any interest.

2. If n is not squarefree, then our algorithm for solving (1) may not work if $1 < (N(M), n) < n$. The reason for this is illustrated by the example $n = t^2$, where t is composite, and $t \nmid M$. Our algorithm might detect the factorization $n = t^2$, and try to use Hensel's lemma to construct a solution modulo t^2 from a solution modulo t . Modulo t , however, the congruence reduces to $X^2 - KY^2 \equiv 0 \pmod{t}$. Without knowing the factorization of t , the only solution we can construct in this case is the trivial one with $X = Y = 0$, and this solution will not work in Hensel's lemma. In fact, Rabin [R] has observed that any algorithm which produces solutions to $X^2 - KY^2 \equiv 0 \pmod{n}$ can be used as a probabilistic algorithm for factoring n . This provides a reason for believing that (1) may be hard to solve if $(N(KM), n) > 1$.
3. In the OSS scheme, K was assumed to be a square modulo n , and part of the secret key used to sign messages was $\sqrt{K} \pmod{n}$. It turns out that this information is not necessary for signing messages in polynomial time.
4. If n is odd, then a solution to (1) exists if $(N(KM), n) = 1$. If n is even, then not all messages M can be signed, even if $(N(KM), n) = 1$. In particular the message $M = \sqrt{d}$ is not signable if K_1 is even, (where $K = K_0 + K_1\sqrt{d}$), so \sqrt{d} is not signable if K is a square. Our method will produce a solution to (1) if such a solution exists.

2. Proof of Theorem 1

The proof of Theorem 1 is elementary, requiring only that we complete the square. To begin, if $1 < (A, n)$ or $1 < (B, n)$ or $1 < (M, n)$ or $1 < (C, n) < n$, then the Euclidean algorithm will produce a nontrivial factor of n . If $1 = (A, n) = (B, n) = (M, n)$ and $n \nmid C$, then solving the congruence in question is equivalent to solving

$$X^2 + BA^{-1}Y^2 \equiv MA^{-1} \pmod{n}.$$

An oracle for Problem I now produces a solution. The only case remaining is if $(A,n) = (B,n) = (C,n) = (M,n) = 1$. By completing the square we get

$$(X + CA^{-1}Y)^2 - [(CA^{-1})^2 - (BA^{-1})]Y^2 \equiv MA^{-1} \pmod{n}. \quad (2)$$

Substituting $Z = Y$ and $W = X + CA^{-1}Y$ gives

$$W^2 - [(CA^{-1})^2 - (BA^{-1})]Z^2 \equiv MA^{-1} \pmod{n}. \quad (3)$$

By assumption $n \nmid [(CA^{-1})^2 - (BA^{-1})]$, so either $(n, C^2 - AB)$ gives a nontrivial factor of n or else an oracle for Problem I produces a solution W, Z to (3). In the latter case, $Y = Z$ and $X = W - CA^{-1}Y$ is a solution to the original congruence.

3. Proof of Theorem 2

If $1 < (M_1, n) < n$, then the Euclidean algorithm gives a nontrivial factor of n , so we may assume that $(M_1, n) = 1$. Since $(N(M), n) = 1$, it follows that M is invertible modulo n , and we can use the Euclidean algorithm to calculate $\bar{M} = \bar{M}_0 + \bar{M}_1\sqrt{d}$ such that $M\bar{M} \equiv 1 \pmod{n}$. If we now want to solve

$$(X_0 + X_1\sqrt{d})^2 - (K_0 + K_1\sqrt{d})(Y_0 + Y_1\sqrt{d})^2 \equiv M_0 + M_1\sqrt{d} \pmod{n}, \quad (4)$$

then it suffices to solve

$$(\bar{M}_0 + \bar{M}_1\sqrt{d})(X_0 + X_1\sqrt{d})^2 - (r_0 + r_1\sqrt{d})(Y_0 + Y_1\sqrt{d})^2 \equiv 1 \pmod{n}, \quad (5)$$

where $r_0 + r_1\sqrt{d} = (\bar{M}_0 + \bar{M}_1\sqrt{d})(K_0 + K_1\sqrt{d})$. Setting $Y_0 = 1$ and $Y_1 = 0$, the left hand side of the congruence (5) becomes

$$\{\bar{M}_0X_0^2 + \bar{M}_0dX_1^2 + 2d\bar{M}_1X_0X_1 - r_0\} + \{\bar{M}_1X_0^2 + \bar{M}_1dX_1^2 + 2\bar{M}_0X_0X_1 - r_1\}\sqrt{d}.$$

By our assumptions we have that $(\bar{M}_1, n) = 1$, $n \nmid \bar{M}_1 d$, and $n \nmid (\bar{M}_0^2 - d\bar{M}_1^2)$. Therefore using an oracle for Problem II, we either get a nontrivial factor of n or a solution X_0, X_1 to the congruence

$$\bar{M}_1 X_0^2 + \bar{M}_1 d X_1^2 + 2\bar{M}_0 X_0 X_1 \equiv r_1 \pmod{n}.$$

Let $c = \bar{M}_0 X_0^2 + \bar{M}_0 d X_1^2 + 2d\bar{M}_1 X_0 X_1 - r_0$. If $(n, c) = 1$, then

$$(X_0 + X_1 \sqrt{d})^2 - (K_0 + K_1 \sqrt{d})(1 + \alpha \sqrt{d}) \equiv c(M_0 + M_1 \sqrt{d}) \pmod{n},$$

giving an output of type a). If $n|c$, then

$$(X_0 + X_1 \sqrt{d})^2 \equiv K_0 + K_1 \sqrt{d} \pmod{n},$$

giving an output of type c). If $1 < (n, c) < n$, then we get a nontrivial factor of n .

4. Proof of Theorem 3

We now show how to solve Problem IV, i.e. how to find solutions of the congruence

$$(X_0 + X_1 \sqrt{d})^2 - (K_0 + K_1 \sqrt{d})(Y_0 + Y_1 \sqrt{d})^2 \equiv M_0 + M_1 \sqrt{d} \pmod{n}, \quad (6)$$

where $(N(KM), n) = 1$. The method uses two appeals to an oracle for Problem III, essentially to replace K and M by integers. There are however several possible outputs from Problem III, and we must show how to solve the congruence (6) in each case.

Let us first observe that if $n = 2^a n_1$, where n_1 is odd, then it suffices to solve the congruence separately modulo 2^a and modulo n_1 , since we can then use the Chinese Remainder Theorem to construct a solution modulo n . In order to construct a solution modulo 2^a when $a \leq 3$, we can simply try all of the finite number of possible values for X and Y .

If $a > 3$, then we first construct a solution modulo 8 (if it exists). We will now show how to use Hensel's lemma to lift the solution modulo 8 to a solution modulo 2^a . Let $X = X_0 + X_1\sqrt{d}$ and $Y = Y_0 + Y_1\sqrt{d}$ be a solution of the congruence $X^2 - KY^2 \equiv M \pmod{2^b}$, where $b \geq 3$. We want to choose $Z, W \in \mathbb{Z}[\sqrt{d}]$ such that

$$(X + 2^{b-1}Z)^2 - K(Y + 2^{b-1}W)^2 \equiv M \pmod{2^{b+1}}. \quad (7)$$

Since $b \geq 3$, this is equivalent to

$$X^2 - KY^2 - M + 2^b(XZ - KYW) \equiv 0 \pmod{2^{b+1}}.$$

Let $X^2 - KY^2 - M = 2^b R$, with $R \in \mathbb{Z}[\sqrt{d}]$. Then it suffices to find Z and W satisfying

$$XZ - KYW \equiv -R \pmod{2}. \quad (8)$$

Since $(N(KM), 2) = 1$, $X^2 \equiv N(X) \pmod{2}$ and $Y^2 \equiv N(Y) \pmod{2}$, it follows that either $(N(X), 2) = 1$ or $(N(KY), 2) = 1$, so that either X or KY is invertible modulo 2. If X is invertible modulo 2, then a solution of (8) is given by $Z = -\overline{X}R$ and $W = 0$. If KY is invertible, then we take $Z = 0$ and $W = \overline{KY}R$. Since (7) is solvable, we can lift the solution modulo 2^b to a solution modulo 2^{b+1} .

It now suffices to show how to solve the congruence (6) in the case n is odd. Consider first the case that $n \nmid d$. In this case (6) reduces to the system of congruences

$$\begin{aligned} X_0^2 - K_0 Y_0^2 &\equiv M_0 \pmod{n} \\ 2X_0 X_1 - K_1 Y_0^2 - 2K_0 Y_0 Y_1 &\equiv M_1 \pmod{n}. \end{aligned}$$

Since $(N(K), n) = 1$ and $(N(M), n) = 1$, it follows that $(K_0, n) = 1$ and $(M_0, n) = 1$, so that an oracle for Problem I will produce a solution X_0, Y_0 to the first of these congruences. Furthermore, at least one of $2X_0$ and $2K_0 Y_0$ will be relatively prime to n since $(M_0, n) = 1$ and n is odd. Hence the second congruence above can be solved using the

Euclidean algorithm.

Next we consider the case n odd and $n \nmid d$. One of the possible outputs from Problem III is a factorization $n = n_1 n_2$. If $(n_1, n_2) = 1$, then we can solve the congruences $x^2 - KY^2 \equiv M \pmod{n_1}$ and combine the results with the Chinese Remainder Theorem to get a solution of (6). This splitting procedure will be required at most $O(\log n)$ times. If in the factorization $n = n_1 n_2$ we have $(n_1, n_2) > 1$, then let $G = (n_1, n_2)$, $n = G^2 H_1$, and $G_1 = (G, H_1)$. If $G_1 = 1$ and $H_1 \neq 1$, then we have a relatively prime factorization and can use the Chinese Remainder Theorem. If $G_1 > 1$, then write $n = G^2 G_1 H_2$, and let $G_2 = (G, H_2)$. Continuing in this manner, since the H_i 's are decreasing, we either arrive at a value $H_i = 1$, or else we find $G_i = 1$ which produces a relatively prime factorization of n . If $H_i = 1$, then it is easy to see that $p|n$ if and only if $p|G$. Hence we can run the algorithm with n replaced by G , and later use Hensel's Lemma to construct a solution modulo a sufficiently large power of G that is divisible by n . It should be remarked that the computations required to apply both Hensel's Lemma and the Chinese Remainder Theorem can be carried out in deterministic polynomial time.

Another possible output from Problem III is a square root of K modulo n . If we know $S \in \mathbb{Z}[\sqrt{d}]$ with $S^2 \equiv K \pmod{n}$, then as in [OSS2] we get the factorization $x^2 - KY^2 \equiv (x - SY)(x + SY)$. It then suffices to solve the linear system

$$x - SY \equiv 1 \pmod{n},$$

$$x + SY \equiv M \pmod{n}.$$

Notice that S is invertible mod n , and also that 2 is invertible mod n since we have assumed that n is odd. Hence the solution to the linear system is provided by

$$x \equiv (M+1)/2 \pmod{n}$$

$$y \equiv (M-1)/(2S) \pmod{n}.$$

We may now disregard the cases in which the output from the oracle for Problem III is not of type a). The first step in solving (6) is to reduce to solving

$$(X + Y\sqrt{d})^2 - (K_0 + K_1\sqrt{d})(W + Z\sqrt{d})^2 \equiv c \pmod{n}, \quad (9)$$

where $c \in \mathbb{Z}$. If $n|M_1$, then (6) is already in the desired form. If $n \nmid M_1$, then use an oracle for Problem III to obtain $X_0, X_1, Y_0, Y_1, c \in \mathbb{Z}$, such that $(c, n) = 1$ and

$$(X_0 + X_1\sqrt{d})^2 - (K_0 + K_1\sqrt{d})(Y_0 + Y_1\sqrt{d})^2 \equiv c(M_0 + M_1\sqrt{d}) \pmod{n}.$$

(The procedure if the oracle returns a type b) or c) output has already been dealt with.) Using an idea from Pollard's original algorithm (see [S] or [PS]) it is now enough to solve (9), since we can use the composition of binary quadratic forms to construct a solution to (6). By the observation of Lenstra (see [OSS2]), the roles of K and c are interchangeable, so to solve (9) it suffices to solve

$$(X + Y\sqrt{d})^2 - c(W + Z\sqrt{d})^2 \equiv (K_0 + K_1\sqrt{d}) \pmod{n}. \quad (10)$$

By the same reasoning that led us to the problem of solving (9), we can use an oracle for Problem III in order to reduce (10) to the problem of solving

$$(X + Y\sqrt{d})^2 - c(W + Z\sqrt{d})^2 \equiv b \pmod{n}, \quad (11)$$

where $b \in \mathbb{Z}$ satisfies $(b, n) = 1$. Finally we use an oracle for Problem I to solve (11) over the rationals.

References

- OSS1 H. Ong, C. P. Schnorr, and A. Shamir, "An Efficient Signature Scheme Based on Quadratic Equations," Proc. 16th ACM Symp. Theor. Comput. (1984) 208-216.
- OSS2 H. Ong, C. P. Schnorr, and A. Shamir, "Efficient Signature Schemes based on Polynomial Equations," to appear In Crypto 84, Lecture Notes in Computer Science, Springer-Verlag, N. Y., 1984.

- PS J. M. Pollard and C.-P. Schnorr, "Solution of $x^2 + ky^2 \equiv m \pmod{n}$, with applications to digital signatures", preprint, 1985.
- SI J. Shallit, "An Exposition of Pollard's Algorithm for Quadratic Congruences," Technical Report 84-006, Department of Computer Science, University of Chicago, Dec. 1984.
- R M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," M.I.T. Laboratory for Computer Science, Technical report LCS/TR-212, 1979.

ANOTHER BIRTHDAY ATTACK

Don Coppersmith
IBM Research
Yorktown Heights, NY 10598

Abstract: We show that a meet-in-the-middle attack can successfully defraud the Davies-Price message authentication scheme. Their scheme used message blocks in an iterated encipherment of an initial block, and it went through the message blocks twice, in order to prevent just such a "birthday" attack.

Background

This note concerns methods for attaching a digital signature to a long message. There are several proposals for hashing the long message into a shorter hashed value, which can then be digitally signed by a more expensive technique, for example RSA. [RSA] This allows the signature to be publicized without revealing the content of the message; it allows a shorter signature; and it decreases the computation time necessary for computing or checking signatures. [Den]

Rabin [Rab] introduced a scheme, based on a general block cipher. It can be described in terms of DES, although Rabin's proposal did not use DES. In this scheme, the message M would be broken into 56-bit blocks M_i , and these message blocks would be used as keys for the iterated encipherment of some initial value H_0 . The final encipherment, along with the initial value, would form the hash value. Thus

$$\begin{aligned}H_0 &= \text{random} \\ H_i &= E_{M_i}(H_{i-1}), \quad 1 \leq i \leq n \\ \text{RSA-Sign}(H_0, H_n).\end{aligned}$$

(Notation: here and throughout, $E_K(X)$ is the DES encipherment of the cleartext X under the key K ; $D_K(Y)$ is the DES decipherment of the ciphertext Y under the key K .)

The problem with this scheme in conjunction with DES, is a "meet-in-the-middle" or "birthday" attack. The opponent, knowing the RSA-signature of the pair (H_0, H_n) arising from some legitimate message M' , can devise a message M whose content is largely selected by the opponent, but whose hash value is also (H_0, H_n) . Thus the RSA-signature of (H_0, H_n) can be reused to sign this bogus message.

To accomplish this, the opponent need only evaluate 2^{33} encipherments, instead of the 2^{64} required by the naive trial-and-error approach. (He also uses $2^{32} = 4 \times 10^9$ storage.) Namely, the opponent specifies values of M_1, M_2, \dots, M_{n-2} . Using the given value of H_0 , he computes successively H_1, H_2, \dots, H_{n-2} . Then for each of 2^{32} trial values X for the message block M_{n-1} , he computes that value $H_{n-1}[X] = E_X(H_{n-2})$ which H_{n-1} would have if X were chosen for M_{n-1} . These 2^{32} values are sorted and stored. Now for each of 2^{32} trial values Y for the message block M_n , he computes that value $H'_{n-1}[Y] = D_Y(H_n)$ which H_{n-1} would need to have in order for H_n to have its correct value, under the assumption that Y were chosen for M_n . Each of these values is compared against the sorted

values for $H_{n-1}[X]$. If a match is found ($H_{n-1}[X] = H'_{n-1}[Y]$) then the assignments $M_{n-1} = X$, $M_n = Y$ complete the message M to one satisfying our requirements. Finally, the expected number of "successful" pairs (X, Y) is 1, so that we will find one with reasonable probability; this probability can be increased by a modest increase in the work factor.

The Davies-Price Scheme

Davies and Price [DP] introduced another DES-based message authentication scheme, by which they hoped to avoid this attack. Their scheme differs from Rabin's in that they cycle through the message blocks twice. Thus,

$$\begin{aligned} H_0 &= \text{random} \\ H_i &= E_{M_i}(H_{i-1}), \quad 1 \leq i \leq n \\ H_{n+i} &= E_{M_{n+i}}(H_{n+i-1}), \quad 1 \leq i \leq n \\ &\text{RSA-Sign}(H_0, H_{2n}). \end{aligned}$$

In the present note, we mount an attack on this scheme, similar to the meet-in-the-middle attack described above, with not much larger computational requirements.

The Attack

Our attack has two phases: a precomputation phase, which can be done once and used against all messages; and a stage tailored to the individual message. The requirements: for the precomputation stage, 2^{37} encipherments and 2^{36} storage; for the individual message, 2^{35} encipherments and 2^{32} storage. There are modest trade-offs available.

The message format is as follows. We select most of the message (say blocks M_{19} through M_n) to be the text of that bogus message which we are trying to authenticate. Blocks M_1 and M_2 are chosen (by a meet-in-the-middle step) to put ourselves into a standardized position. Finally, blocks M_3, M_4, \dots, M_{18} are chosen, from among possibilities enumerated during the precomputation, to "meet in the middle" one last time.

During the precomputation, we select an arbitrary 64-bit quantity Z , which is going to be the value of H_2, H_4, H_6, \dots , and H_{18} . We select 2^{36} trial values X , compute the values $E_X(Z)$, and sort and store these values. Now select 2^{36} trial values Y , compute the values $D_Y(Z)$, and compare each against the values $E_X(Z)$. Record each match: $E_X(Z) = D_Y(Z)$. We expect to find about 256 such pairs $\{(X_i, Y_i), 1 \leq i \leq 256\}$; if not, examine a few more values of Y . Each such pair (X_i, Y_i) can be used as a message pair $(M_3, M_4), (M_5, M_6), \dots$, or (M_{17}, M_{18}) , in the sense that if we have $H_2 = Z$, $M_3 = X_i$, $M_4 = Y_i$, then we get $H_4 = Z$.

Given a message $M' = (M_{19}, M_{20}, \dots, M_n)$, an RSA-signature of some pair (H_0, H_{2n}) , the chosen value of Z and the 256 pairs (X_i, Y_i) gotten during precomputation, our task is to select values of M_1, M_2, \dots, M_{18} which will make (H_0, H_{2n}) a valid hash of $M = (M_1, M_2, \dots, M_n)$.

First we find values of M_1 and M_2 such that $E_{M_1}(H_0) = D_{M_2}(Z)$; this takes 2^{33} encipherments and 2^{32} storage. We know that $H_2 = Z$, so as long as the pairs $(M_3, M_4), \dots, (M_{17}, M_{18})$ are chosen from our list (X_i, Y_i) , we will have $H_4 = H_6 = \dots = H_{18} = Z$. Assuming $H_{18} = Z$, use the values M_{19} through M_n to compute the value H_n ; with the

values M_1 and M_2 we can then get the value of H_{n+2} . Working backwards from H_{2n} , using the values $M_n, M_{n-1}, \dots, M_{19}$, we find the value of H_{n+18} .

Now we use the precomputed pairs (X_i, Y_i) . For each of $256^4 = 2^{32}$ choices of four pairs (X_i, Y_i) to be the values of $(M_3, M_4), (M_5, M_6), (M_7, M_8), (M_9, M_{10})$, compute the value of H_{n+10} that would result. (The efficient way to do this is to run through the pairs lexicographically, so as not to recompute $E_{X_i}(H_{n+2})$ for each of 2^{24} occurrences.) Sort and store these trial values of H_{n+10} . Similarly, select pairs to be the values of $(M_{11}, M_{12}), (M_{13}, M_{14}), (M_{15}, M_{16}), (M_{17}, M_{18})$, and compute backwards from H_{n+18} to get trial values of H_{n+10} . Compare against the stored trial values. We expect one match, and the corresponding values of M_3 through M_{18} finish our task.

Extensions

The Davies-Price scheme could be altered by running through the message three times instead of twice. This attack will still work, at the expense of a large increase in the number of "constrained" message blocks (the message blocks chosen by the algorithm, rather than selected by the user).

Another possible scheme would be to set up two initializing vectors.

$$\begin{aligned} H_0, H'_0 &= \text{random} \\ H_i &= E_{M_i}(H_{i-1}), \quad 1 \leq i \leq n \\ H'_i &= E_{M'_i}(H'_{i-1}), \quad 1 \leq i \leq n \\ \text{RSA-Sign}(H_0, H_n, H'_0, H'_n). \end{aligned}$$

Minor modifications to the present attack allow this scheme to be broken as well. Namely, do the same precomputation as before, and compute M_1, M_2 as before. Work forwards to find H_{n-2} , then use a meet-in-the-middle step to discover values M_{n-1}, M_n which satisfy the requirement on H_n . Then the values M_3 through M_{18} can be selected as before (from the pairs (X_i, Y_i)) to satisfy the requirements on H'_n .

A word about "constrained" message blocks: since we only need to examine $2^{36} < 23^8$ values X in the precomputation, we can select them to be EBCDIC representations of alphanumeric characters, so that even the "constrained" message blocks needn't look like total nonsense. In fact, at the risk of increasing the number of such blocks, we can increase their plausibility, to the point of having a set English text, with the freedom of choice made by substitution of synonyms. [DP]

Trade-offs

The presentation here tried to minimize computation time. There are two trade-offs available, which increase the computation time but decrease (1) storage and (2) length of constrained message, respectively.

When running a meet-in-the-middle attack, we work forward with J values, sort and store the outcomes; work backwards with K values, and compare against the J values stored. We are likely to succeed if $JK \geq N$, where N is the size of the space (in our case 2^{64}). Thus we trade off storage of J against computation time of K , subject to $K \geq J, JK \geq N$.

In the present attack, we had 18 blocks of constrained message. This can be decreased if we are willing to spend more time in precomputation. A precomputation of 2^{41} encipherments and 2^{40} temporary storage would allow us to recover $2^{16} = 65536$ pairs (X_i, Y_i) , and with that larger selection we would need to add only ten constrained message blocks: two at the beginning as before, and four pairs $(M_3, M_4), \dots, (M_9, M_{10})$ to allow the last meet-in-the-middle step to go through ($65536^4 = 2^{64} = N$.)

References

- [Den] D.E. Denning, Protecting public keys and signature keys, *IEEE Computer*, 1983, 16(2):27.
- [DP] D.W. Davies and W.L. Price, "The Application of Digital Signatures based on Public Key Cryptosystems," NPL Report DNACS 39/80, National Physical Laboratory, Teddington, Middlesex, England, Dec. 1980.
- [Rab] M. Rabin, Digital Signatures, in "Foundations of Secure Computation," Academic Press, New York, 1978.
- [RSA] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM, vol. 21, no. 2, Feb. 1978, pp. 120-126.

ATTACKS ON SOME RSA SIGNATURES

Wiebren de Jonge¹ and David Chaum²

¹Department of Mathematics and Computer Science
Vrije Universiteit P.O. Box 7161
1007 MC Amsterdam The Netherlands

²Centre for Mathematics and Computer Science
Kruislaan 413
1098 SJ Amsterdam The Netherlands

ABSTRACT

Two simple redundancy schemes are shown to be inadequate in securing RSA signatures against attacks based on multiplicative properties. The schemes generalize the requirement that each valid message starts or ends with a fixed number of zero bits. Even though only messages with proper redundancy are signed, forgers are able to construct signatures on messages of their choice.

1. INTRODUCTION

The basic notions of redundancy in signatures and multiplicative attacks are introduced for completeness in this introductory section, along with an example which is used in subsequent sections. Next the two redundancy schemes are presented briefly. An algorithm is then described and used to construct attacks on the two schemes. Finally, a second kind of attack is presented which also compromises the two redundancy schemes.

1.1. THE NEED FOR REDUNDANCY

RSA used in its raw form does not protect against a forger choosing an integer S_c with $0 < S_c < n_A$, and computing $M_c = (S_c)^{e_A} \bmod n_A$ from it, where n_A and e_A are A 's public modulus and exponent in an RSA system. Subsequently, the forger could claim that S_c is the signature on M_c . Since exponentiation modulo n acts as a kind of one-way function when $\phi(n)$ is

unknown, this *chosen signature attack* can be used for finding signatures on “random” (i.e., unpredictable) messages only. Thus, it may be said that only the signer can form signatures on chosen messages, but anybody can determine which message corresponds to a chosen signature.

To prevent these unpredictable messages from having a reasonable chance of being accepted, *redundancy* will be required in signed messages. Hence, a distinction will be made between *messages* and *valid messages*: all numbers M with $0 \leq M < n$ are messages, but only a very small fraction of these will be valid messages. For instance, if 100 bits of redundancy are used, a chosen signature will have only a chance of 2^{-100} of corresponding to a valid message. Thus, finding a *false signature* (i.e., a signature on a valid message not actually signed by A) will cost 2^{99} trials on the average, which makes it infeasible to successfully guess a signature.

Some work has been based on the assumption that the signer would sign anything except some desired messages. ([DeMillo & Merritt 82] and [Denning 83] independently generalized and extended [Davida 82].) Under these assumptions, attackers were able to obtain signatures on desired messages simply by combining signatures on apparently unrelated messages. The seemingly more realistic and practical model assumed here, that the signer is only willing to sign valid messages, makes attacks more difficult—though not impossible—as will be shown.

1.2. MULTIPLICATIVE ATTACKS

Preventing chosen signature attacks not only requires a sufficient *quantity* of redundancy in valid messages; it also necessitates that the *nature* of the redundancy is appropriate, since RSA signatures are *multiplicative*.

For example, suppose that B can construct three valid messages M_1 , M_2 and M_3 such that $M_3 = (M_1 \cdot M_2) \bmod n_A$. Then, if B succeeds in getting M_1 and M_2 signed by A , B can form the product (modulo n_A) of these signatures to get a false signature on M_3 , denoted $S_A(M)$, since

$$\begin{aligned} S_A(M_3) &= (M_1 \cdot M_2)^{d_A} \bmod n_A \\ &= ((M_1^{d_A} \bmod n_A) \cdot (M_2^{d_A} \bmod n_A)) \bmod n_A \\ &= (S_A(M_1) \cdot S_A(M_2)) \bmod n_A. \end{aligned}$$

B can also use the inverse M^{-1} or the opposite $-M$ of a message M , assuming the corresponding signed version is known, as a factor in a product forming a new message, since $S_A(M^{-1} \bmod n_A) = (S_A(M))^{-1} \bmod n_A$ and $S_A(-M) = -S_A(M)$. (Notice that d_A is odd.)

Thus, if B knows A 's signature on one or more valid messages M_i , B can easily forge signatures for valid messages that B can rewrite as a product of message(s) M_i , their opposite(s) $-M_i$,

or their inverse(s) M_i^{-1} (all in modulo n_A arithmetic). Note also that a message and/or its opposite and/or its inverse may occur in a product more than once. Therefore, the redundancy should make it infeasible to find such valid messages.

1.3. EXAMPLE CRYPTOSYSTEM

Rivest, Shamir and Adleman recommended that n be about 200 decimal digits, which amounts to about 664 bits [RSA 78]. We will use a particular example of an RSA system for illustrative purposes, in which n is 800 bits, thereby maintaining an ample margin of safety with respect to known factorization techniques for appropriate moduli. The amount of redundancy used in the examples will be 200 bits. One reason for this choice of amount of redundancy is to provide for sufficient protection against a chosen signature attack. Another is for efficiency, since one does not want to expand the messages to be signed too much, say, not more than one third. Since, for our choice of n , RSA limits signed messages to 800 bits, the redundancy should amount to at most 200 bits. As a consequence, only a fraction of 2^{-200} of all 800-bit messages are valid, and thus the original message to be signed, called the *actual* message, may comprise 600 bits.

An important assumption is that every bit pattern of 600 bits represents a meaningful message; the only redundancy is that explicitly included in the remaining 200 bits.

2. THE TWO REDUNDANCY SCHEMES

In the first redundancy scheme the redundant bits are combined with the actual message by multiplying that message with an agreed on constant w . That is, all messages M for which $M \bmod w = 0$ are defined to be valid. The actual message present in such a valid message is $m = M \div w$. For $w = 2^{200}$, this means that each valid message ends up with 200 zero bits.

In analogy to the special case where w is a power of two, the general scheme will be called the *right-padded redundancy* scheme. Figure 1 shows how the right-padded redundancy scheme spreads the valid messages over the interval $[0, n]$ for $n=91$ and $w=6$.

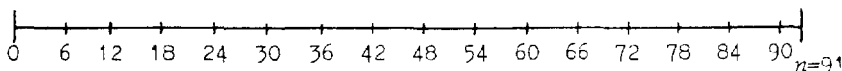


Fig. 1. The valid messages in case of right-padded redundancy for $n=91$ and $w=6$.
The valid messages are 0, 6, 12, ..., 90.
The actual messages are 0, 1, 2, ..., 15.

The counterpart of spreading valid messages over the interval $[0, n]$ is to concentrate them in some interval $[l, u]$ of appropriate size. The actual message contained in a valid message M then is $m = M - l$. For $l=0$ and $u=2^{600}-1$, each valid message starts with a sequence of 200 zero bits. Accordingly, we call the general case of this scheme *left-padded redundancy*. Figure 2 illustrates the left-padded redundancy scheme for $n=91$, $l=19$ and $u=34$.

3. A FIRST KIND OF ATTACK

Before treating our attacks in detail, we mention briefly an algorithm that will be used heavily.

3.1. A VARIATION OF EUCLID'S ALGORITHM

The algorithm finds, for a given x with $0 \leq x < n$, the smallest positive value c such that $(cx) \bmod n$ is less than some given threshold value t . It is very similar to Euclid's algorithm for computing the greatest common divisor. Indeed, Euclid's algorithm can be used to compute an increasing sequence of values c for which the corresponding values $(cx) \bmod n$ form a decreasing sequence. The only important difference is that processing with our algorithm stops as soon as a value below the given threshold is reached. Since Euclid's algorithm has a worst case (and average case) complexity of $O(\log n)$ [Knuth 69], our algorithm will certainly be fast (enough) too.

For our purposes, it is often important that the value found for c is reasonably small. Although our algorithm can be used to find for any given values x and t the smallest c for which $(cx) \bmod n \leq t$, there is no guarantee that c itself is smaller than some other threshold value. However, it is easy to show that there always exists some c with $0 < |c| \leq n/t$ for which $0 \leq (cx) \bmod n \leq t$.

Consider the integers $a/$ and $b/$ such that $(ax - b) \bmod n$ with $0 < a \leq \lceil n/t \rceil$ and $0 \leq b \leq t$. Since there are more than n different pairs (a, b) , there exist two different pairs, say, (a_1, b_1) and (a_2, b_2) , for which $(a_1x - b_1) \bmod n = (a_2x - b_2) \bmod n$. Since x usually will be co-prime with n (if not, one could factor n), we know that both $a_1 \neq a_2$ and $b_1 \neq b_2$. Therefore, we may safely assume that $b_1 > b_2$. Thus, for $c = (a_1 - a_2)$ it is true that $0 < |c| \leq n/t$ and



Fig. 2. The valid messages with left-padded redundancy for $n=91$, $l=19$ and $u=34$.
The valid messages are 19, 20, 21, ..., 34.
The actual messages are 0, 1, 2, ..., 15.

$$0 < (cx) \bmod n = (b_1 - b_2) < t.$$

Since our algorithm searches for the smallest positive value c for which $(cx) \bmod n \leq t$, the value found for c may be larger than n/t . If so, the above shows that there exists some c , $0 < c \leq n/t$, such that $0 \leq (-cx) \bmod n \leq t$. This c can be found by applying our algorithm to $(-x) \bmod n$.

3.3. ATTACKING RIGHT-PADDED REDUNDANCY

If RSA is used in combination with the right-padded redundancy scheme, one attack proceeds as follows. First, choose an actual message m_1 (i.e., $m_1 < n_A \text{ div } w$) on which A 's signature is desired. An attack, such as this, allowing a signature to be constructed for a chosen actual message will be called a *chosen message attack*. Now, $M_1 = m_1 w$ is a valid message, since $M_1 < n_A$ and $M_1 \bmod w = 0$. Next, compute $x = (m_1 w)^{-1} \bmod n_A$. If $w \leq n^{1/2}$, i.e., if the redundancy takes up less than half of the bits in a valid message, our algorithm of Figure 3 can be used to find a number $0 < c \leq n_A \text{ div } w$ such that $(cx) \bmod n_A \leq n_A \text{ div } w$ or $(-cx) \bmod n_A \leq n_A \text{ div } w$. Thus, one can find two actual messages m_2 and m_3 such that $m_2 = (m_3 x) \bmod n_A$ or $m_2 = (-m_3 x) \bmod n_A$.

If one succeeds in getting A 's signature on m_2 and m_3 (i.e., $S_A(M_2)$ and $S_A(M_3)$), one can compute A 's signature on m_1 by multiplying $S_A(M_3)$ with the inverse of $S_A(M_2)$ and taking the opposite in the case we used $-x$. Naturally, all arithmetic is done modulo n_A . In case we used just x , this works, because

$$\begin{aligned} S_A(M_1) &= S_A((x^{-1} \cdot (m_3 w) \cdot (m_3 w)^{-1}) \bmod n_A) \\ &= (S_A(m_3 w) \cdot S_A((m_3 x w)^{-1})) \bmod n_A \\ &= (S_A(M_3) \cdot (S_A(M_2))^{-1}) \bmod n_A; \end{aligned}$$

in case $-x$, we have

$$S_A(M_1) = (S_A(M_3) \cdot (-S_A(M_2))^{-1}) \bmod n_A.$$

Of course, the attack makes sense only when $m_1 \neq m_2$ and $m_1 \neq m_3$. But if the found m_2 or m_3 happens to be equal to m_1 , one simply searches for another value of c such that $(cx) \bmod n \leq t$ or $(-cx) \bmod n \leq t$. For example, one tries the next minimal value of $(cx) \bmod n$ or $(-cx) \bmod n$, respectively.

3.4. ATTACKING LEFT-PADDED REDUNDANCY

RSA's multiplicative properties are also useful for attacking the RSA signature system when left-padded redundancy is used. Recall that this scheme defines valid messages as those in the interval $[l, u]$.

As a first step it will be shown why, in the general case, l should be larger than $u^{1/2}$, and thus in our example should be larger than 2^{300} . If l would be smaller than $u^{1/2}$, then any two valid messages M_i ($i=1,2$) out of $[l, u^{1/2}]$ have a product, say M_3 , which lies in the interval $[l, u]$. This makes a multiplicative attack far too easy. Thus, the left-padded redundancy scheme should certainly not be used with $l=0$; i.e., just requiring each valid message to start with a certain number of zero-bits immediately appears to be unsuitable.

For $l > u^{1/2}$ there is a chosen message attack. Suppose that M is the valid message on which a false signature is desired. First, the attack will be shown for $M \leq u-l$, and later it will be extended for the more likely case that $M > u-l$.

Due to the large number of wrap-arounds, the number $(l \cdot M) \bmod n$ may be positioned anywhere in the interval $[0, n]$. Therefore, the chance that $l \cdot M \bmod n$ lies in the interval $[l, u]$ is negligibly small. (About 2^{-200} in the example.) However, it is easy to find a positive integer i such that $(l+i)M \bmod n$ is in $[l, u]$.

For example, suppose we have the situation as depicted in Figure 3, where $l \cdot M \bmod n$ is positioned somewhere to the right of $[l, u]$. Clearly, $(l+1)M \bmod n$ lies a (relatively small) step of size M to the right of $l \cdot M \bmod n$, $(l+2)M \bmod n$ lies another such step further to the right, and so on. Thus, it is easy to compute i , the number of steps to the right needed to end up in the "next" interval $[l, u]$. Since M is supposed to be less than $u-l$, the step size is small enough to prevent the interval from being missed by jumping too far.

Thus, if $l+i$ happens to be in $[l, u]$, we have found three valid messages M_i ($i=1,2,3$) with $M_1 = l+i$, $M_2 = M$ and $M_3 = (l+i)M \bmod n$ for which $M_3 = (M_1 M_2) \bmod n$. Thus, a false signature on M can be constructed from the signatures on M_1 and M_3 .

To be sure that $l+i$ indeed will be in $[l, u]$, i.e., that $i \leq s = u-l$, the step size should be large enough, i.e., $M \geq n/s$. Because of our assumption that $M \leq s = u-l$, and the interval size s should be larger than n/s . Therefore, this attack works for all chosen messages M with

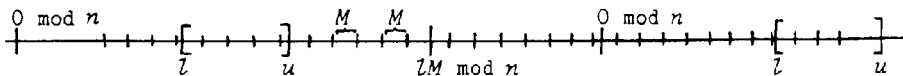


Fig. 3. An illustration of the basic idea of the attack.
Note that this figure is not drawn to scale!

$n/s \leq M \leq s$ if $s \geq n^{1/2}$, i.e., if the redundancy takes up less than half of the bits in a valid message.

If $M > s = u - l$, there is no guarantee that a "walk" to the right with steps of size M will end up in the "next" interval $[l, u]$. For example, if $u - l = 2^{600}$ and if $M \approx 2^{700}$ then the chance to hit the next first interval $[l, u]$ on a walk to the right is only about 2^{-100} .

However, as explained in §3.1, it is easy to find a value c for which $|c| \leq n/s$ and $cM \bmod n \leq s$. Starting with $x = l$ or $x = u$, one can use this new value $cM \bmod n$ as the step size (to the right or to the left) and can compute for which integer i the number $(x + ic)M \bmod n$ will be in the interval $[l, u]$. Since we want $x + ic$ to be a valid message, the product ic should be less than s . Assuming that each number less than s had equal probability of being the chosen step size $cM \bmod n$, the chance that the step size is, for a given p , larger than s/p is $1 - 1/p$. With a step size larger than s/p , i will be less than $(np)/s$. Thus, ic will then be less than $(n^2p)/s^2$. This upper bound on ic should be kept smaller than s , therefore, s should be such that $s^3 > (n^2p)$. In other words, the chance of success is very large roughly when the redundancy is less than one third of the bits in a valid message.

Consider our example with $l = 2^{700}$. The number c for which $cM \bmod n \leq 2^{600}$ or $(-cM) \bmod n \leq 2^{600}$ will be less than 2^{200} . There is a high probability that the new step size, $cM \bmod n$ or $(-cM) \bmod n$, will be larger than, say, 2^{500} . This means that the required number of steps, i , almost certainly will be less than 2^{300} . In our example, ic thus may be expected to be less than 2^{500} . This means that we could have started with almost any x in $[l, u]$.

4. A SECOND STYLE OF ATTACK

Another kind of attack is based on an approach called Multiplying-In-Dividing-Out (MIDO). It is used below to break the same two redundancy schemes.

4.1. RIGHT-PADDED REDUNDANCY AGAIN

Suppose that the actual message m on which a false signature is desired, can be written as the product of two numbers a_1 and a_2 . Thus, $M = mw = a_1 a_2 w < n$. Now choose numbers b_1 and b_2 such that $M_1 = a_1 b_1 w < n$, $M_2 = a_2 b_2 w < n$, and $M_3 = b_1 b_2 w < n$ (e.g., choose any b_1 and b_2 with $b_1 < a_2$ and $b_2 < a_1$). Clearly, the three messages M_1 , M_2 and M_3 are all valid, and $M = \frac{M_1 M_2}{M_3}$ (hence the name MIDO). Thus, if one succeeds in getting A 's signature on the valid messages M_i ($i = 1, 2, 3$), one can also construct a false signature on the chosen message M .

One difference with the attack of §3.3 is that this MIDO attack works for any amount of redundancy. On the other hand, this MIDO attack will not work for all chosen messages m , since it may be infeasible or even impossible to factor the integer m . Of course, one could mani-

pulate chosen factors to construct an appropriate actual message m , but this does not change the fact that there is only limited freedom in choosing m .

4.2. LEFT-PADDED REDUNDANCY REVISITED

The following method illustrates how the MIDO approach can be used for attacking left-padded redundancy. It works for all valid messages M that can be written as a product $a_1 a_2$ with $a_1 \geq a_2 \geq 2$, such that $a_1 \neq a_2 + 1$ and either (a) $M - l > a_2$ and $u - M > a_1$ or (b) $M - l > a_1$ and $u - M > a_2$.

In case condition (a) holds, take

$$\begin{aligned} M_1 &= (a_1 - 1)a_2 = M - a_2, \\ M_2 &= a_1(a_2 + 1) = M + a_1, \\ \text{and } M_3 &= (a_1 - 1)(a_2 + 1) = M + a_1 - a_2 - 1. \end{aligned}$$

Thus, $M = \frac{M_1 M_2}{M_3}$, while condition (a) assures that all three messages M_i ($i=1,2,3$) are valid.

The condition $a_1 \neq a_2 + 1$ assures that $M_3 \neq M$. For the case that condition (b) is true, a_1 and a_2 should be exchanged in the above description. Figure 4 illustrates how M_1 , M_2 and M_3 are positioned in $[l, u]$ if condition (a) holds.

Clearly, the chance of success with this method depends on the size and placement of the interval $[l, u]$, and thus on the amount of redundancy. Furthermore, this method does not work for all chosen messages. However, it is easy to adapt this attack to work for almost any chosen valid message M . The only restriction will be that M should not be chosen too close to l or u . Such a restriction is not very severe, since, for example, $u - M$ and $M - l$ are both larger than $2^{-10}(u - l)$ for 99.8 percent of all valid messages.

Once M is chosen, one searches for "factors" a_1 and a_2 such that $M = (a_1 a_2) \bmod n$. (The important difference with the attack above is the addition of "mod n ".) This can easily be accomplished by freely choosing one factor, say a_1 , and then computing the other factor, a_2 , as $(a_1^{-1} M) \bmod n$. Having fixed a_1 and a_2 one computes the numbers c_1 and c_2 with $|c_1| \leq 2n / (u - M)$ and $|c_2| \leq 2n / (M - l)$ such that $(c_1 a_1) \bmod n < (u - M) / 2$ and $(c_2 a_2) \bmod n < (M - l) / 2$. In the following, we only treat the case that both c_1 and c_2 are positive. Take

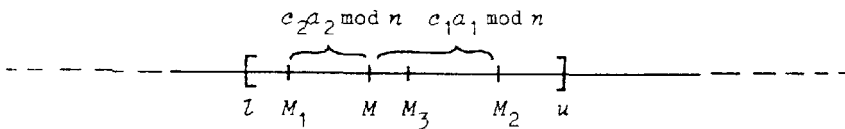


Fig. 4.

$$\begin{aligned}
M_1 &= a_2(a_1 - c_2) \bmod n = M - (c_2 a_2 \bmod n) \\
M_2 &= a_1(a_2 + c_1) \bmod n = M + (c_1 a_1 \bmod n) \\
\text{and } M_3 &= (a_1 - c_2)(a_2 + c_1) \bmod n = (M + c_1 a_1 - c_2 a_2 - c_1 c_2) \bmod n.
\end{aligned}$$

Thus, M_1 and M_2 are valid messages. Define z to be the minimum of $u - M$ and $M - l$. M_3 is also a valid message if $c_1 c_2$ is appropriately small, i.e., if $c_1 c_2$ is less than $z/2$. (See Figure 5 for an illustration.) Since $c_1 c_2$ is known to be less than $4n^2/z^2$, this product is certainly smaller than $z/2$ if $8n^2 < z^3$. Thus, the attack works essentially when the redundancy amounts to less than one third of a valid message.

In our example, both $u - M$ and $M - l$ are numbers of almost 600 bits. Therefore, c_1 and c_2 may be expected to be numbers of a good 200 bits. Thus, their product may be estimated to be a number of something like 400 bits, which usually will be negligibly small compared to M 's distance to l and u . As a consequence, the chance that M_3 is not in the interval $[l, u]$ is negligibly small.

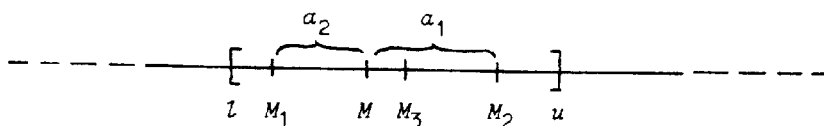


Fig. 5.

CONCLUDING REMARKS

The attacks presented use signatures obtained on messages having a redundancy property that are chosen to allow derivation of false signatures on other messages also having the redundancy property. The attacks are quite powerful, since they allow the derived message to be chosen freely or almost freely.

One obvious way to protect against attacks such as those shown here in practice, which has been known in the "folklore" of cryptography for some time, is to apply some sort of one-way function to actual messages before signing them. This approach can be quite practical for long messages. But for short messages, it may have the disadvantage of data expansion and may be unnecessarily computationally expensive.

There are of course signature schemes that do not appear to have the kinds of multiplicative structures used in the attacks presented here. These schemes generally have received less attention than RSA and most of those currently unbroken appear more expensive than RSA in various ways. An interesting and potentially attractive variation on RSA signatures, however, came out of this work [de Jonge 85].

Multiplicative properties of RSA and its variants should not necessarily be regarded as undesirable shortcomings to be avoided in improved systems, however, since they allow various

powerful and often desirable functionality, such as blind signatures [Chaum 85]. Motivation for embarking on this line of inquiry in fact came from consideration of the needs for secure blind signature systems. In such systems, any message may be signed; only messages with the redundancy property are accepted; and the primary security requirement, called conservation of signatures, is that it should not be possible to construct more signatures than are issued. Thus such systems do require redundancy properties robust in the presence of multiplicativity. The simple schemes considered here demonstrate that such redundancy properties must be chosen with care.

ACKNOWLEDGEMENTS

We are grateful to Evert Wattel and Jan-Hendrik Evertse for some stimulating discussions.

REFERENCES

- (1) Chaum, D., "Security Without Identification: Transaction Systems to make Big Brother Obsolete," *Communications of the ACM*, Vol. 22, No. 10, October 1985, pp. 1030-1044.
- (2) Davida, G.I., "Chosen Signature Cryptanalysis of the RSA (MIT) Public Key Cryptosystem," Technical Report TR-CS-82-2, University of Wisconsin, Milwaukee WI, October 1982.
- (3) de Jonge, W., "Attacks on RSA Signatures and Countermeasures," in *Security and Privacy in Information Systems: some technical aspects*, Ph.D. Thesis, June 1985.
- (4) DeMillo, R.A. and Merritt, M.J., "Chosen Signature Cryptanalysis of Public Key Cryptosystems," Technical Memorandum, School of Information and Computer Science, Georgia Institute of Technology, Atlanta GA, October 25, 1982.
- (5) Denning, D.E., "The Many-Time Pad: Theme and Variations" *Proceedings of the 1983 Symposium on Security and Privacy*, April 25-27, 1983; the relevant part also appeared as "Digital Signatures with RSA and Other Public-Key Cryptosystems," *Communications of the ACM*, Vol. 27, No. 4, April 1984, pp. 388-392.
- (6) Knuth, D.E., *The art of computer programming*, Volume 2, *Seminumerical Algorithms*, Addison-Wesley, 1969.
- (7) Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, February 1978, pp. 120-126.

AN ATTACK ON A SIGNATURE SCHEME *
PROPOSED BY OKAMOTO AND SHIRAISHI

Ernest F. Brickell
Bell Communications Research
Morristown, NJ 07960

and

John M. DeLaurentis
Sandia National Laboratories
Albuquerque, NM 87185

Abstract

Recently Okamoto and Shiraishi proposed a public key authentication system [1]. The security of the scheme is based on the difficulty of solving quadratic inequalities. This new system is interesting since the amount of computing needed for the proposed scheme is significantly less than that needed for an RSA encryption.

This report is an investigation into the security of the proposed digital signature scheme. We demonstrate that if the system is used as it is presented, an opponent could sign messages without factoring the modulus. Further, we suggest a modification which may not have the same flaw as the proposed scheme.

Introduction

Prior to the publication of this authentication system, Ong, Schnorr, and Shamir presented a public key signature scheme [2] which was based on the difficulty of solving a quadratic equation over the ring of integers modulo n (here n is the product of two large rational primes). Pollard produced a random polynomial time algorithm [3] which would allow an opponent to sign messages without knowing the secret key. In an attempt to overcome the weakness pointed out by

* This work performed at Sandia National Laboratories supported by the U. S. Dept. of Energy under contract No. DE-AC04-76DP00789.

Pollard, a new version of the signature scheme was introduced [4]. This variant was based on the difficulty of solving a polynomial equation over the quadratic integers. It has been shown that the new system is also insecure [5]. In fact, breaking the latest scheme can be "reduced" to the problem of solving the original quadratic equation.

The digital signature scheme proposed by Okamoto and Shiraishi is similar to the ones proposed by Ong, Schnorr, and Shamir in that it is based on the difficulty of solving a quadratic expression. More precisely, the signature s is considered to be valid for the message m if and only if

$$(*) \quad h(m) \leq s^2 \pmod{n} \leq h(m) + \delta, \quad \delta = O(n^{2/3})$$

and s is not "small in absolute value"; that is, $\gamma \leq s \leq n-\gamma$, for a suitably chosen γ . Here $h(\cdot)$ is a "one-way" function and the modulus n has the form $n = p^2q$, for large primes p, q . In this paper we will use the expression $x \pmod{n}$ to denote the least nonnegative integer congruent to $x \pmod{n}$. The idea behind the authentication scheme is to force an opponent to compute an approximate square root for $h(m)$.

Cryptanalysis of the Basic Scheme

We show that an opponent can sign messages without knowing the factorization of n by using the following procedure: Choose x such that for some positive integers k, l and nonnegative integer c we have

$$2kx = ln + c$$

where $k = O(n^{1/12})$ and $c = O(n^{1/6})$. (For example, let $x = \lceil l/2k \rceil$, $k = O(n^{1/12})$, $c = O(n^{1/6})$.) Next we calculate

$$y = (h(m) - x^2) \pmod{n},$$

$$z = (k^{-2}y) \pmod{n},$$

$$z = \lceil \sqrt{z} \rceil = \sqrt{z} + \epsilon,$$

here $\lceil w \rceil$ is the least integer which is greater than or equal to w . Finally we set

$$s = x + ka.$$

We sign the message m with s . To verify that s satisfies condition (*) notice that

$$\begin{aligned}
 s^2 \pmod{n} &\equiv (x^2 + 2kxa + k^2a^2) \pmod{n} \\
 &\equiv (x^2 + ca + k^2(z + 2\sqrt{z}\epsilon + \epsilon^2)) \pmod{n} \\
 &\equiv (x^2 + ca + y + 2k^2\sqrt{z}\epsilon + k^2\epsilon^2) \pmod{n} \\
 &\equiv (h(m) + ca + 2k^2\sqrt{z}\epsilon + k^2\epsilon^2) \pmod{n} \\
 &= h(m) + \delta
 \end{aligned}$$

where $\delta = O(n^{2/3})$ as desired.

This ensures that the signature s would be accepted as authentic.

Cryptoanalysis of the Lower Bits Method

Okamoto and Shiraishi proposed another signature scheme which they call the lower bits method. In addition to the modulus $n = p^2q$ and the one-way function h , they add ϵ to the public key where ϵ is an integer and $\epsilon = O(n^{1/3})$. s is considered a valid signature of m if and only if for $s' = (s^2 - h(m)) \pmod{n}$ and s' the least nonnegative residue, either

$$s' \equiv 0 \pmod{\epsilon}$$

or

$$(n - s') \equiv 0 \pmod{\epsilon},$$

and s is again not "small in absolute value."

An opponent can forge messages if he can take square roots mod ϵ , which he can do if he knows the factorization of ϵ . To forge a signature to m , pick x such that for some positive integers k , λ and non-negative integer c

$$2kx = \lambda n + c$$

where $k^2\epsilon^2 + c\epsilon < n$. Next calculate

$$x' = h(m) - x^2 \pmod{n}$$

and a such that $0 < a < \epsilon$ and

$$k^2 a^2 + ca \equiv x' \pmod{\epsilon}.$$

Let $s = x + ka$. Then

$$\begin{aligned} s^2 - h(m) &\equiv x^2 + 2kxa + k^2 a^2 - h(m) \pmod{n} \\ &\equiv x^2 - h(m) + k^2 a^2 + ca \pmod{n} \\ &\equiv x^2 - h(m) + f\epsilon + x' \pmod{n} \\ &\equiv f\epsilon \pmod{n}. \end{aligned}$$

Since

$$0 < k^2 a^2 + ca, x' < n$$

then

$$-n < f\epsilon < n.$$

Hence if $s' = s^2 - h(m) \pmod{n}$ (i.e., s' is the least nonnegative residue), then either

$$s' = f\epsilon \quad (\text{if } f > 0)$$

or

$$n - s' = -f\epsilon \quad (\text{if } f < 0).$$

A Secure (?) Modification

Suppose that instead of signing messages with approximate square roots, the designer chose to sign messages with k^{th} roots, i.e., s is a signature for m whenever $s^k \equiv h(m) \pmod{n}$, $k > 4$. To be more precise, the signature s is considered valid if the following inequality holds

$$(**) \quad h(m) < s^k \pmod{n} < h(m) + \delta, \quad \delta = O(n^{2/3}).$$

The legitimate user can sign messages in nearly the same fashion as in the original scheme. Pick a random $x \in \mathbb{Z}_{pq}^*$ (\mathbb{Z}_{pq}^* is the multiplicative group modulo pq). Compute s as follows

$$s = x + ypq$$

where

$$y = w(kx^{k-1})^{-1} \pmod{p}$$

and

$$w = \{[h(m) - x^k \pmod{n}]/(pq)\}.$$

It can be shown that s satisfies (**). We do not know if the modified scheme possesses the same flaw as the original system. However, in view of the demonstrated weakness of the Okamoto-Shiraishi quadratic inequality scheme and the unsuccessful attempts made by Ong, Schnorr and Shamir, the security of the modified system is highly questionable.

References

- [1] T. Okamoto, A. Shiraishi, "A Fast Signature Scheme Based on Quadratic Inequalities," Proc. of the 1985 Symposium on Security and Privacy, April 1985, Oakland, CA.
- [2] H. Ong, C. P. Schnorr, and A. Shamir, "An Efficient Signature Scheme Based on Quadratic Equations," Proc. 16th ACM Symp. Theor. Computing (1984), 208-216.
- [3] J. M. Pollard, "Solution of $x^2 - ky^2 \equiv m \pmod{n}$," Private communication with C. P. Schnorr, June 29, 1984.
- [4] H. Ong, C. P. Schnorr, and A. Shamir, "Efficient Signature Schemes Based on Polynomial Equations," to appear in Crypto'84, Lecture Notes in Computer Science, Springer-Verlag, NY (1984).
- [5] D. Estes, L. Adleman, K. Kompella, K. McCurley, G. Miller, "Breaking the Ong-Schnorr-Shamir Signature Scheme for Quadratic Number Fields," to appear.

A SECURE SUBLIMINAL CHANNEL (?)*

Gustavus J. Simmons
Sandia National Laboratories
Applied Mathematics Department
Albuquerque, New Mexico 87185

Introduction

At Crypto'83, the present author showed that a transmitter and chosen receiver(s) -- by secretly exchanging some side information -- could pervert an authentication without secrecy channel to allow them to convert a portion of the authentication information to a hidden (covert) communications channel [1]. It was also shown that under quite reasonable conditions even the detection of the existence of this covert channel could be made as difficult as the underlying authentication algorithm was "cryptosecure". In view of this open -- but undetectable -- existence, such a covert channel was called a "subliminal" channel. The examples constructed in [1] were more in the nature of existence proofs than of practical subliminal communications channels. At Eurocrypt'84 [2], however, it was shown how to use digital signature schemes as a way of realizing practical subliminal channels and, in particular, subliminal channels were devised using Ong and Schnorr's quadratic approximation scheme [3], Ong, Schnorr and Shamir's quadratic representation schemes [4] and Ong, Schnorr and Shamir's cubic signature scheme [5] as well as Gamal's discrete logarithm-based digital signature scheme [6]. Unfortunately, from the standpoint of providing a secure (and feasible) subliminal channel, all of these digital signature schemes were cryptanalyzed [7,8] shortly after being proposed. At Crypto'84, a fourth variant to the earlier digital signature schemes of Ong, Schnorr and Shamir was presented by Schnorr [9] which was also quickly cryptanalyzed [10]. At the 1985 IEEE Symposium on Security and Privacy, Okamoto and Shiraishi proposed yet another digital signature scheme based on quadratic inequalities [11] which had been designed to avoid the cryptanalytic weaknesses that had flawed the schemes of Schnorr, et al. The cryptanalysis of this scheme by Brickell and DeLaurentis is reported elsewhere in these Proceedings [12]. In view of the short-lived nature of all of these schemes, it has become a high risk venture to propose subliminal channels based on digital signatures. The motivation for doing so is that digital signatures can be much easier to calculate and verify than full-fledged two-key ciphers. As a result, the benefits (of a successful implementation) far outweigh

* This work performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract no. DE-AC04-76DP00789.

the risks of perhaps having an insecure digital signature (or subliminal) channel slip by undetected. Based on the cumulative experience gained in cryptanalyzing the six digital signature schemes mentioned above, Brickell and DeLaurentis propose a new scheme in their paper that appears to avoid the weaknesses exploited in the earlier cryptanalyses.

It is an easy matter to adapt the Brickell-DeLaurentis digital signature scheme to accommodate a subliminal channel, however the resulting channel has a protocol weakness, common to all of the subliminal channels thus far devised, that we wish to avoid. In this paper we first point out the nature of this weakness and then propose a modified form of the Brickell-DeLaurentis digital signature scheme in which a subliminal channel can be embedded -- free of the protocol weakness.

The Protocol Weakness (Problem)

The problem is that in all subliminal channels devised thus far, the subliminal receiver -- by virtue of the side information that must be given to him by the transmitter to enable him to recover the subliminal communications -- is in a privileged position to impersonate the transmitter. In other words, the transmitter and subliminal receiver have to be mutually trusting and trustworthy parties. There are, of course, some applications in which this is the case, but in general the transmitter prefers that the ability to receive subliminal communications not be synonymous with an ability to forge undetectable signatures in his stead. Since the same protocol weakness runs through all of subliminal schemes, we illustrate it using the channel which we proposed at Eurocrypt 84 based on the Ong, Schnorr and Shamir quadratic representation digital signature scheme [2,4]. In the interest of both completeness and brevity we summarize the essential points in their scheme for the three steps: key generation, signature generation and signature verification.

Key Generation

1. Tx chooses a composite n which is computationally infeasible to factor. The factorization of n is kept secret (if known).
2. Tx chooses a random u , $(u, n) = 1$, and calculates $k = -u^{-2} \pmod{n}$. u is kept secret.
3. Tx publishes n and k as his authentication key.

Signature Generation

Given a message m , $(m, n) = 1$, to be "signed":

1. Tx chooses a random r , $(r, n) = 1$. r is kept secret.
2. Tx calculates

$$s_1 = \frac{1}{2} \left(\frac{m}{r} + r \right) \pmod{n}$$

$$s_2 = \frac{u}{2} \left(\frac{m}{r} - r \right) \pmod{n}$$

3. The triple $(m; s_1, s_2)$ is transmitted as the "signed" message.

Authentication of Signature

1. Rx receives $(m; s_1, s_2)$

2. Rx calculates

$$a \equiv s_1^2 + k \cdot s_2^2 \pmod{n}$$

3. The message m is accepted as authentic if and only if

$$a = m$$

To set up the subliminal channel, in addition to the steps taken by the transmitter in the key generation procedure for the digital signature scheme, the transmitter secretly communicates u to the designated receiver, Rx^\dagger , for the subliminal channel. Now, when the transmitter wishes to send a signed message m through the overt channel and a covert message m^* through the subliminal channel, where it is still desired that both the Rx^\dagger and third parties be able to verify the authenticity of the signature to m , the transmitter generates the signature as follows.

Signature Generation for the Subliminal/Signature Channel

Given a message m , $(m, n) = 1$, to be "signed" and a message m^* , $(m^*, n) = 1$, to be communicated subliminally:

1. Tx calculates

$$s_1 = \frac{1}{2} \left(\frac{m}{m^*} + m^* \right) \pmod{n}$$

$$s_2 = \frac{u}{2} \left(\frac{m}{m^*} - m^* \right) \pmod{n}$$

2. The triple $(m; s_1, s_2)$ is transmitted as the "signed" message.

Authentication of the signature by either the subliminal receiver, Rx^\dagger , or by third parties is unaffected by the presence of the subliminal communication. The subliminal receiver, however, knowing u can solve for the subliminal message as follows:

Decoding the Subliminal Message

The subliminal Rx^\dagger , given $(m; s_1, s_2)$ and knowing u , calculates

$$m^* = \frac{m}{s_1 + s_2 u^{-1}} \pmod{n}$$

to recover the covert message m^* "hidden" by the Tx in the signature of m .

Since the subliminal transmitter and receiver share the same piece of secret information, u , they are clearly interchangeable in terms of their capabilities. This is also true of the subliminal channel based on the Brickell-DeLaurentis digital signature scheme. In the next section, we show how to avoid this serious protocol failure in a subliminal channel embedded in a digital signature scheme similar to the one proposed by Brickell and DeLaurentis.

The Secure Subliminal Channel (?)

We borrow from Brickell and DeLaurentis the notion of basing the cryptosecurity of a digital signature on the difficulty of extracting approximate k^{th} roots in Z_n , n composite. While $n = p^2q$ in their scheme, we require $n = p^2qr$ for reasons that will become apparent later; p, q and r are all appropriately chosen primes $p > q$ and $q > r$. Again, in the interest of brevity, we summarize the essential points involved in signing messages using the modified Brickell-DeLaurentis digital signature scheme.

Key Generation

1. Tx chooses three primes $p > q > r$ sufficiently large that p^2q is computationally infeasible to factor. p, q and r are kept secret.
2. Tx publishes $n = p^2qr$ as his authentication key. The receivers need to know (or calculate) a bound $\delta = O(n^{2/3})$. The Tx may choose to treat δ as a redundant part of the key.
3. Both the Tx and $Rx(s)$ know a one-way hashing function on messages, $h(m): m \in Z_n, h(m) \in Z_n$ and an exponent $k \geq 4$.

Signature Generation

Given a message m , $m \in Z_n$, to be "signed":

1. Tx chooses a random $x \in Z_{pqr}^*$ (Z_{pqr}^* is the set of integers less than pqr and relatively prime to q , p and r).
2. Tx first calculates the one-way hashing function $h(m)$, and then calculates the signature s of m as follows:

$$a. \quad w = \left[\frac{h(m) - x^k \pmod{n}}{pqr} \right]$$

$$b. \quad y = \frac{w}{kx^{k-1}} \pmod{p}$$

$$c. \quad s = x + y pqr \quad .$$

3. The pair $(m; s)$ is transmitted as the "signed" message.

Authentication of Signature

1. Rx receives $(m; s)$.
2. Rx calculates the hashing function $h(m)$.
3. The message is accepted as authentic if and only if

$$(1) \quad h(m) \leq s^k \pmod{n} < h(m) + \delta$$

In the Appendix we show that an s (signature) generated according to this protocol satisfies (1).

This modification of the Brickell-DeLaurentis scheme is at least as cryptosecure as their scheme. If these schemes turn out to be cryptosecure, this modification leads to the simplest subliminal channel yet devised. The transmitter secretly gives to the intended subliminal receiver(s) the prime r . Once this has been done, subliminal communication takes place as follows.

Signature Generation for the Subliminal/Signature Channel

Given a message $m \in Z_n$ to be "signed" and another message $m^* \in Z_r$ to be communicated subliminally:

1. Tx calculates s using m^* . He chooses a random $u \in Z_{pq}^*$ and calculates

$$x^* = m^* + ur$$

which is used instead of a random $x \in Z_{pqr}^*$ to calculate s as before.

Any receiver, including the subliminal receiver(s), Rx^\dagger , can authenticate a message exactly as before, but in addition Rx^\dagger can recover m^* .

Decoding the Subliminal Message

1. Rx^\dagger , given $(m; s)$ and knowing r calculates

$$s = x^* + ypqr = m^* + ur + ypqr \equiv m^* \pmod{r}$$

On the other hand, since one needs to know pqr in order to sign messages, a subliminal receiver -- knowing only r and $n = p^2qr$ -- needs to factor p^2q in order to recover pqr . It thus appears that this subliminal channel is just as cryptosecure to a subliminal receiver attempting to impersonate the transmitter as the Brickell-DeLaurentis scheme is secure to an outsider attack.

Incidentally, if the same message were signed repeatedly, using either this scheme or in the Brickell-DeLaurentis scheme, a random appearing set of signatures would result.

Appendix

As in the discussion of a secure subliminal channel, let the modulus n be of the form

$$n = p^2qr \quad p > q > r \quad \text{all primes}$$

and $M \in \mathbb{Z}_n$, $s \in \mathbb{Z}_n^*$.

Theorem:

$$(1) \quad M \leq s^k \pmod{n} < M + pqr$$

if and only if

$$(2) \quad s = x + ypqr$$

$$(3) \quad y = \frac{w}{kx^{k-1}} \pmod{p}$$

$$(4) \quad w = \frac{(M - x^k \pmod{n})}{pqr}$$

where $x \in \mathbb{Z}_{pqr}^*$, $y \in \mathbb{Z}_p$ and $w \in \mathbb{Z}_p$.

Proof:

First, assume that (1) holds. We show that (2), (3) and (4) follow.

Given $s \in \mathbb{Z}_n^*$, s has a unique representation of the form

$$s = x + ypq$$

where

$$x \in \mathbb{Z}_{pqr}^* \quad \text{and} \quad y \in \mathbb{Z}_p.$$

x and y are given by

$$s \equiv x \pmod{pq}$$

and

$$y = \left\lfloor \frac{s}{pqr} \right\rfloor$$

respectively. Now form

$$s^k = x^k + kx^{k-1}ypq + p^2q^2r^2 \times (\text{higher order terms})$$

$$s^k \equiv x^k + kx^{k-1}ypq \pmod{n}.$$

Now since (1) was satisfied by hypothesis

$$M \leq s^k \pmod{n} = x^k + kx^{k-1}ypq < M + pqr$$

we have

$$\frac{M - x^k \pmod{n}}{pqr} \leq kx^{k-1}y < \frac{M + pqr - x^k \pmod{n}}{pqr}$$

or

$$kx^{k-1}y = \left[\frac{M - x^k \pmod{n}}{pqr} \right] = w$$

and

$$y = \frac{w}{kx^{k-1}}.$$

Next, assume that (2), (3) and (4) hold, then

$$s^k = x^k + kx^{k-1}ypqr + p^2q^2r^2 \text{ (HOT)}$$

$$s^k \equiv x^k + kx^{k-1}ypqr \pmod{n}.$$

Replacing y by $y = \frac{w}{kx^{k-1}}$ we obtain,

$$s^k \equiv x^k + wpqr \pmod{n}$$

and finally,

$$s^k \equiv x^k + \frac{M - x^k \pmod{n}}{pqr} pqr \pmod{n}$$

from which (1) is an easy consequence.

References

1. G. J. Simmons, "The Prisoners' Problem and the Subliminal Channel," Proceedings of Crypto'83, Santa Barbara, CA, Aug. 21-24, 1983, in Advances in Cryptology, Ed. by D. Chaum, Plenum Press, New York (1984), pp. 51-67.
2. G. J. Simmons, "The Subliminal Channel and Digital Signatures," Proceedings of Eurocrypt'84, to appear.
3. H. Ong and C. P. Schnorr, "Signatures through Approximate Representations by Quadratic Forms," Proceedings of Crypto'83, Santa Barbara, CA, August 21-24, 1983, to be published by Plenum Press.
4. H. Ong, C. P. Schnorr and A. Shamir, "An Efficient Signature Scheme Based on Quadratic Equations," Proceedings of 16th Symposium on Theory of computing, Washington D.C., April 1984, to appear.
5. C. P. Schnorr, "A Cubic OSS-Signature Scheme," private communication, May 1984.

6. T. El Gamal, "A New Public Key Cryptosystem and Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, to appear.
7. J. M. Pollard, "Solution of $x^2 - KY^2 \equiv m \pmod{n}$," Letter to Schnorr, 29/6/84.
8. J. Shallit, "An Exposition of Pollard's Algorithm for Quadratic Congruences," Technical Report 84-006, Department of Computer Science, University of Chicago, Dec. 1984.
9. H. Ong, C. P. Schnorr, and A. Shamir, "Efficient Signature Schemes Based on Polynomial Equations," to appear in *Crypto'84, Lecture Notes in Computer Science*, Springer-Verlag, NY (1984).
10. D. Estes, L. Adleman, K. Kompella, K. McCurley, G. Miller, "Breaking the Ong-Schnorr-Shamir Signature Scheme for Quadratic Number Fields," to appear.
11. T. Okamoto, A. Shiraishi, "A Fast Signature Scheme Based on Quadratic Inequalities," Proc. of the 1985 Symposium on Security and Privacy, April 1985, Oakland, CA.
12. E. Brickell and J. DeLaurentis, "An Attack on a Signature Scheme Proposed by Okamoto and Shiraishi," these Proceedings.

UNCONDITIONALLY SECURE AUTHENTICATION SCHEMES AND PRACTICAL AND THEORETICAL CONSEQUENCES

Yvo Desmedt¹

Dept. of Computer Science², University of New Mexico
Albuquerque, New Mexico, U.S.A.

current address:

Katholieke Universiteit Leuven, ESAT01
Kardinaal Mercierlaan, 94, B-3030 Heverlee, Belgium

Abstract

The Vernam scheme protects the privacy unconditionally, but is completely insecure to protect the authenticity of a message. Schemes will be discussed in this paper that protect the authenticity unconditionally. The definition of unconditional security is defined. Stream cipher authentication schemes are proposed. The consequences on information protection using RSA and DES are discussed.

1. Introduction

We will start here by looking how some authors discuss the protection of authenticity in a conventional cryptosystem. The definitions given for unconditional security will be overviewed. We will conclude that both subject matters are mostly presented oversimplified. This will be explained by checking their definitions using the Vernam scheme (see Section 2). We will conclude that unconditionally authentication protection is not discussed. Hereto we define unconditional security from a point of view of authenticity and we also redefine the old definition of an unconditionally secure cryptosystem (see Section 3). We will then build up an unconditionally secure authentication system (see Section 4). The practical and theoretical consequences will be presented (see Section 5).

Some authors, e.g., Denning [4], (pp. 10) pretend that "*in symmetric (conventional) cryptosystems ... secrecy cannot be separated from authenticity*", and that "*if users cannot access E_A and D_A , then both the secrecy and authenticity of A 's data is assured*". However, today it is well-known that one can authenticate the message (and the sender) without protecting the privacy (of the whole message (see the previous last paragraph of Section 4.3 and Section 6.1)). The NBS authentication method [11] (pp. 24) is an example of this. It is also known that some modes as e.g., the E.C.B. mode in DES, are insecure to protect the authenticity of a *long* message. As Diffie and Hellman [8] (pp. 646) said: "*A cryptographic system intended to guarantee privacy will not, in general, prevent this latter*

¹NFWO aangesteld navorser, is currently sponsored by the National Science Foundation of Belgium.

²This research was done while the author was Visiting Assistant Professor at the University of New Mexico.

form of mischief". One can conclude that Denning's ideas, earlier cited, are at least oversimplified. One can wonder if *each cryptosystem which protects the privacy can also authenticate long messages if one uses modes* (e.g. CFB or CBC).

The term unconditionally secure is misleading. One can have the impression that it covers more. When one says that the one-time pad cryptosystem is unconditionally secure, one can think that one can never attack the privacy or authenticity. The definition or use of unconditional security only deals today with privacy protection (see e.g. [4], [16], [17], [22]). As Simmons remarked in [22], Shannon's models [21] were only concerned with secrecy. One can wonder if *a scheme which protects the privacy unconditionally does the same for the authenticity*.

We will now answer both questions by discussing the Vernam (or one-time pad) cryptosystem [21], [24].

Important remark

If in the following sections we say that an intruder can inject a fraudulent message with a probability p_1 or that an active eavesdropper can modify a message with a probability p_2 , we mean the following: in one on $1/p_1$ respectively $1/p_2$ cases the system used by the receiver will not automatically detect the injection respectively the modification. Automatically here means that the system uses a different way to detect modifications in the message than using the redundancy in the language. If the reader does not agree with this restriction we remark that the worst case is a message without redundancy. In order to be able to deal with such messages previous restriction is evident.

2. The Vernam scheme and authentication

As known, the Vernam scheme protects the privacy unconditionally [21]. Let us shortly explain how it works. Let $M = (m_1, m_2, \dots, m_n)$ be the plaintext message, where m_1 is the first bit of the message, m_2 the second, and so on. Then the ciphertext $C = (c_1, c_2, \dots, c_n)$ is the bitwise exor of M and the key K , or $c_i = m_i \oplus k_i$. The key K is really random and only used once. The decryption operation is similar: $m_i = c_i \oplus k_i$.

It is now easy to understand that the probability to inject a fraudulent bit is $1/2$ (see important remark in Section 1). If the active eavesdropper wants that the receiver receives a bit 1, he injects a bit (it does not matter if it is a zero or a one that he injects). Because the key bit is in one on two cases (in average) a 0 (and otherwise a 1), the receiver receives a 1 in one on two cases. One can remark that the effect of this attack is not important, because if one wants that the receiver accepts a concrete fraudulent message of 100 bits, the probability to succeed by injecting a message is only $1/2^{100}$. However, in some cases the damage caused by the injecting of one bit may be important. That one bit may tell you to delete or not to delete a file, to transfer the money or not.

An even more serious attack is to *modify* the ciphertext. It is easy to understand that an active eavesdropper can modify a bit of the plaintext with a probability 1. Hereto he has only to complement the ciphertext bit. In the case the active eavesdropper does not know the plaintext, the effect of his action will probably be a not understandable message (sometimes called "garbage"). However, for terrorists that does not matter, it is enough to sabotage. If the active eavesdropper *knows the plaintext*, he can easily modify it as he wants (if the fraudulent message is not longer than the original one)!

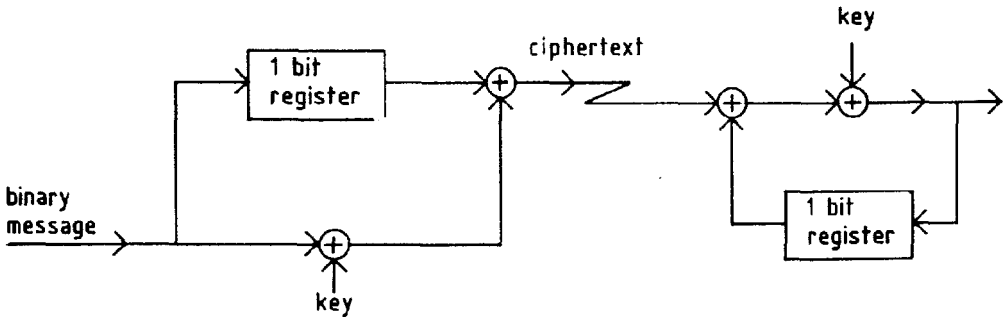


Figure 1: The Vernam scheme used in a CBC mode

We can conclude that the Vernam scheme can not protect the authenticity. A similar remark was made by Feistel [9] (pp. 19 – 20), without coming up with an unconditionally secure authentication scheme. However, one could remark that, to protect the authenticity of a message, one has to use a mode and some redundancy at the end of the message (e.g. 64 zeros). In order to show that this does not help, we first suppose that the sender uses the CBC mode in feedforward and the receiver uses it in feedback (in order to have a large error propagation). The plaintext is followed by 64 zeros as authenticator. Using the Vernam scheme for the encryption and decryption devices we obtain Figure 1. The active eavesdropper can modify each bit of the message as he wants, without affecting the authenticator! Suppose he wants to modify m_i . His attack is successful, if he complements bits c_i and c_{i+1} (where c_i is the i^{th} transmitted ciphertext bit). If the active eavesdropper wants to modify more bits, he has to superimpose previous attack. If the sender and receiver do not want to protect the privacy and use hereto a similar scheme as the NBS one [11], the attack is similar, because the Vernam scheme encrypts bit after bit (its block length is one bit). The reader can easily extend the attack for other modes as CFB and OFB (see [11]).

We have shown that the Vernam scheme can not protect the authenticity. Schemes which protect the privacy do not necessarily protect the authenticity, even if some modes are used. This result can be extended to schemes which are similar (e.g. the Vignere scheme).

We will now try to come up with cryptosystems which protect the authenticity unconditionally. Evidently, we have first to define what means an unconditionally secure authentication scheme.

3. New definitions for unconditional security

Definition 1: A cryptosystem protects the privacy unconditionally if, no matter how much ciphertext is intercepted, there is not enough information in the ciphertext to come

up with a unique solution (plaintext), but many exist. Ideal and perfect cryptosystems [21] fall under this definition.

Remark that:

- the fact of unconditional security can not be affected by using more (e.g. an infinite amount of) computertime.
- if the appropriate security rules are satisfied (e.g. secrecy of the key) the cryptosystem can *never* be broken!

We now want to come up with a similar definition for the unconditional protection of the authenticity (of message and sender). Let us start from the last remark. So we would say that a cryptosystem protects unconditionally the authenticity if it can never be broken. However, then no cryptosystem at all would satisfy this definition. The process of authentication is probabilistic. If an active eavesdropper tries long enough (e.g. some centuries) he will be able to inject a fraudulent message or modify one. This follows from the fact that messages have finite lengths. So a better definition, based on the first remark, will now be given.

Definition 2: A cryptosystem protects the authenticity unconditionally with a security level P if, the probabilities that an intruder can inject a fraudulent message or that an active eavesdropper can modify a message are less or equal than $1/P$, independently how much computertime is used. If one of these probabilities is equal to one we say that the system is insecure (to protect the authenticity).

As a consequence of this definition the Vernam scheme is insecure related to authenticity. A scheme is considered to be *insecure for practical purposes* if the security level is "too" small. We will not discuss what means "too" small, the reader is referred to [5] and [6].

The effect of birthday attacks [1], [14] (pp. 127) is not discussed in this paper.

Let us now build cryptosystems which satisfy last definition.

4. Building unconditionally secure authentication schemes

Based on the analysis of the Vernam scheme we will first try to come up with cryptosystems which protect the authenticity unconditionally. Because the schemes which will be proposed in Section 4.1, do not satisfy partly or totally the definition, we will in Section 4.2 come up with a secure one. In Section 4.3 a more practical version will be discussed. Finally by combining other schemes with the ones discussed here, one can still improve the practical aspects (see Section 4.4).

4.1. Trials

We have seen that the Vernam scheme is bit oriented. In the first proposal we use an authenticator for each bit. Each bit of the message is followed by a *fixed pattern* (e.g. 64 zeros). All these bits are then encrypted using the Vernam scheme. We now give a formal way to describe this scheme. If the plaintext $M = (m_1, m_2, \dots, m_n)$ then the input for the Vernam scheme is $A = (a_1^1, a_1^2, \dots, a_1^q, a_2^1, a_2^2, \dots, a_2^q, \dots, a_n^1, a_n^2, \dots, a_n^q)$, where a_i^j are bits such that for all i ($1 \leq i \leq n$) we have $(a_i^1, a_i^2, \dots, a_i^q) = (m_i, 0, \dots, 0)$ in the case the fixed pattern is $(0, 0, \dots, 0)$ and $q - 1 = \text{length of the fixed pattern}$. The

ciphertext is then $C = (c_1^1, c_1^2, \dots, c_1^q, c_2^1, c_2^2, \dots, c_2^q, \dots, c_n^1, c_n^2, \dots, c_n^q)$ where $c_i^j = a_i^j \oplus k_i^j$ and the key is $K = (k_1^1, k_1^2, \dots, k_1^q, k_2^1, k_2^2, \dots, k_2^q, \dots, k_n^1, k_n^2, \dots, k_n^q)$, where k_i^j is a bit (for all i and j , where $1 \leq i \leq n$ and $1 \leq j \leq q$). Remark that we have an expansion of the ciphertext and the key with a factor q (e.g. $\times 65$). This scheme is however insecure (from the point of view of authenticity) because an active eavesdropper can complement a ciphertext bit, corresponding with an information bit, without modifying the ciphertext bits corresponding with that authenticator (in other words complementing c_i^1 , without modifying any c_i^j where j satisfies $2 \leq j \leq q$). This implies that the probability that an active eavesdropper can modify a message is evidently one.

In another proposal the information bit is placed at random in the authenticator. The mathematical description of this scheme is similar, except that $(a_1^1, a_1^2, \dots, a_1^q) = (0, 0, \dots, 0, m_i, 0, \dots, 0, 0)$, where the bit m_i appears on a random location. In order that the receiver could verify that location he has to know the random value (otherwise an active eavesdropper can easily modify a message bit 0 into a message bit 1). This random value is a part of the *authentication key*. We define the authentication key as the key which is used to protect the authenticity, and the privacy key as the key which is used to protect the privacy. The length of the authentication key is $n \cdot \lceil \log_2 q \rceil$. The length of the privacy key is $q \cdot n$ and the expansion of the ciphertext is q . The probability that an active eavesdropper can modify a message bit is however high and is $1/q$. Evidently, if the active eavesdropper knows q and modifies one bit of the q bits randomly, the probability to modify the bit corresponding with a certain m_i is $1/q$. In order to obtain a *practical acceptable security level* (e.g. 2^{64}) the expansion of the ciphertext has to be enormous, because the security level only increases linearly with increasing expansion of the ciphertext.

The last scheme will be modified to come up with an unconditionally secure authentication scheme, for which the security level increases exponentially with linearly increasing ciphertext and key expansion.

4.2. A secure scheme

By studying previous proposal we see that the security level is q , and there exist q different $(a_i^1, a_i^2, \dots, a_i^q)$ for each $m_i = 1$. In general, we can have $2^q - 1$ different $(a_i^1, a_i^2, \dots, a_i^q) \neq (0, 0, \dots, 0)$. Hereto we use an authentication key $H = (h_1^1, h_1^2, \dots, h_1^q, h_2^1, h_2^2, \dots, h_2^q, \dots, h_n^1, h_n^2, \dots, h_n^q)$ random such that for all i : $(h_i^1, h_i^2, \dots, h_i^q) \neq (0, 0, \dots, 0)$ and the key H is secret and used only once by sender and receiver (similar as in the Vernam scheme). The scheme differs only from previous one in the fact that for each bit m_i : $(a_i^1, a_i^2, \dots, a_i^q) = m_i \cdot (h_i^1, h_i^2, \dots, h_i^q)$. In other words if $m_i = 0$ then $(a_i^1, a_i^2, \dots, a_i^q) = (0, 0, \dots, 0)$, otherwise $(a_i^1, a_i^2, \dots, a_i^q) = (h_i^1, h_i^2, \dots, h_i^q)$.

Remark that in the discussed scheme the ciphertext expansion is still q . The length of the authentication key is $q \cdot n$, and the same for the privacy key. So the complete key used in this scheme is $2q$ times longer than in the Vernam scheme.

The discussed scheme protects the authenticity unconditionally with a security level $2^q - 1$. An intruder can inject a bit 0 with a probability $1/2^q$, because in order to inject a 0 (after that the legitimate $(i - 1)^{\text{th}}$ bit was sent) he has to guess the correct $(k_i^1, k_i^2, \dots, k_i^q)$ and because these bits are really random he has only a probability $1/2^q$ to succeed. A similar reasoning is true for the case he wants to inject a 1 (remark that $(h_i^1 \oplus k_i^1, h_i^2 \oplus k_i^2, \dots, h_i^q \oplus k_i^q) \neq (k_i^1, k_i^2, \dots, k_i^q)$ for all i). So he can inject a bit with a probability $1/2^{q-1}$. An active eavesdropper can modify a bit with a probability $1/(2^q - 1)$, because he has

to guess correctly $(h_i^1, h_i^2, \dots, h_i^q)$. Remark that it is "hard" for an active eavesdropper to mix the bits of the plaintext, or to retransmit them, because the key is really random and only used once. This follows easily from previous discussion. Remark that previous discussions remain valid if we consider known plaintext attacks, as long as the privacy and authentication keys are secret.

In order to better understand the discussed scheme let us wonder what happens if we do not protect the privacy (or if $(k_1^1, k_1^2, \dots, k_1^q, k_2^1, k_2^2, \dots, k_2^q, \dots, k_n^1, k_n^2, \dots, k_n^q) = (0, 0, \dots, 0)$). The reader can easily verify that the injection of a bit 0 or the modification of a plaintext bit 1 into a 0 is easy. However, it is "hard" to inject a bit 1 or to modify a plaintext bit 0 into a 1. We can conclude that the protection of the privacy is necessary in order to protect, with this scheme, the authenticity. However, one can easily imagine situations in which the protection of a bit 1 is more crucial than the protection of a bit 0 [5], [6]. In E.F.T. for example, the plaintext can be a bit 1 if the transaction is authorized, a 0 in the other cases. Following our definition of unconditional security we do not consider the scheme secure under these circumstances. One can wonder if the key H has to be secret. The answer is evidently yes, otherwise an active eavesdropper can easily modify the message.

Without discussing if this scheme is practical (see Section 5.1) we can remark that such a large text expansion is impractical. It slows down the communication and makes it much more expensive! For these reasons a more practical scheme will now be presented.

4.3. A more practical scheme

An unconditionally secure authentication system which is based on the one discussed in previous section, will be presented. For previous scheme, remark that if an intruder wants to inject two bits the probability to succeed is $1/2^{2q-2}$. In general it is for m bits $1/2^{mq-m}$, because each bit has its own authenticator. A similar reasoning is true for the modification of m bits. *In this section we will only use a authenticator for the whole message.* That idea will also solve the speed and cost problem of previous scheme. Nevertheless the new scheme is also unconditionally secure.

In this scheme we send the message M enciphered with the Vernam scheme, followed by an authenticator of q bits. So the ciphertext is $C = (c_1, c_2, \dots, c_n, c_{n+1}^1, c_{n+1}^2, \dots, c_{n+1}^q)$ such that for $i < n+1$ we have $c_i = m_i \oplus k_i$, where c_i , m_i and k_i are bits. For $i = n+1$ we have $c_{n+1}^j = r^j \oplus k_{n+1}^j$ for each j such that $1 \leq j \leq q$, where c_{n+1}^j and k_{n+1}^j are bits. $R = (r^1, r^2, \dots, r^q)$ is the authenticator and $K' = (k_1, k_2, \dots, k_n, k_{n+1}^1, k_{n+1}^2, \dots, k_{n+1}^q)$ is the privacy key. Remark that c_i^j has no sense here if $i < n+1$, similar for the key and for the message. The register R is build up iteratively when each message bit m_i is sent, using the authenticator key $H = (h_1^1, h_1^2, \dots, h_1^q, h_2^1, h_2^2, \dots, h_2^q, \dots, h_n^1, h_n^2, \dots, h_n^q)$. The contents of R in the begin is 0, then $(r^1, r^2, \dots, r^q) := (r^1 \oplus m_i h_i^1, r^2 \oplus m_i h_i^2, \dots, r^q \oplus m_i h_i^q)$ for each m_i , where $1 \leq i \leq n$. In other words at the end

$$r^j = \bigoplus_{i=1}^n m_i h_i^j \quad \text{for each } j \quad (1 \leq j \leq q). \quad (1)$$

This scheme is unconditionally secure with a security level $(2^q - 1)$. An intruder can inject a message that will be accepted with a probability $1/2^q$, because only one on 2^q messages give that authenticator R . An active eavesdropper can only modify one bit of

the message with a probability $1/(2^q - 1)$ to succeed, because he has to guess the correct $(h_i^1, h_i^2, \dots, h_i^q)$ if he wants to modify m_i . If he wants to modify more bits, he has to guess the correct modification (see Eqn. 1), the probability to succeed is only about $1/2^q$.

In order to better understand this scheme, let us wonder if we need to protect the privacy. If we do not protect the privacy (or if K' is equal to zero), then the previous reasoning remains valid, except that it is easy to inject the message $(0, 0, \dots, 0)$ or to modify a message into that zero message. Indeed, for a zero message the authenticator R is zero. However a very simple protocol can overcome the transmission of a zero message. One could for example agree that if (e.g.) the first bit of the message is one, the real message is zero. If the first bit of the message is zero then the message is not zero. With such a protocol the pattern $(0, 0, \dots, 0)$ will never be sent and as a consequence the authenticity of the message can be protected without protecting the privacy. Remark that the authentication key H has to be secret in all circumstances otherwise modification is easy.

The discussed scheme can be used to protect the authenticity of a message without protecting the privacy. However this system is not acceptable in countries or in circumstances that "others" want to be able to verify that the communication is not used for spying. Such situations can occur as a restriction of local laws, or to be used to verify military actions, e.g., a ban of the testing of nuclear weapons [22]. The reason, why the described algorithm is unacceptable is that one can understand the message M , but one is never sure that the sender will not transmit a secret message instead of the authenticator R .

The length of the key in these schemes is $(q + 1)n$ bits respectively $qn + n + q$ bits, depending if we only protect the authenticity, or privacy and authenticity. The keyexpansion is only the half compared with the scheme discussed in Section 4.2. The ciphertext expansion here is $(n + q)/n$ or not significant. Now, a scheme will be presented in which the length of the key is only about the double of the length of the message (about $2n$ bits). Remark that in a practical secure scheme q is normally 64, such that the expansion of the key in the scheme we just discussed, is still large.

4.4. Other unconditionally secure authentication schemes

Some other authors discussed unconditionally secure authentication schemes before, but did not use this name. Simmons [23] and Brickell [2] discussed several bounds related to the security level, the keylength, etc. They called a system *perfect* (or *double perfect*) if the key was used optimally, or was not longer than necessary. Gilbert et. al. [12] discussed implementations of such perfect authentication systems.

It is easy to prove that the schemes discussed in previous sections are not perfect in the sense defined by Simmons [23] or double perfect as defined by Brickell [2]. This means that the key is not used optimally. To obtain such an optimal keylength one could use projective planes, as discussed in [12] on pp. 414 - 415. However for long messages (e.g. Megabits) the calculations in the Gilbert scheme are awful. Now a scheme will be presented which is unconditionally secure, for which the calculations are not too awful, and for which the expansion of the key is only about two.

The idea is that the users first agree on a lowerbound for the security level P . The message is divided up in blocks of length $q = \lceil \log_2 P \rceil$ bits. So the message $M = (M_1, M_2, \dots, M_\alpha)$ where $\alpha \cdot q \geq n$ and $(\alpha - 1) \cdot q < n$. If n is not a multiple of q then one fills

the message up with zeros. The security level will be 2^q . For each q bits a key of length $2q$ bits is used. So the length of the total key is $2\alpha q$ (about $2n$) bits and is really random. The idea of projective planes [12] is used to generate for each M_i a binary vector $(t_i^1, t_i^2, \dots, t_i^q)$ in $GF(2^q)$ (remark that this binary vector was called c on page 414 in [12]). The scheme continues as the previous one (see Section 4.3) except that:

- instead of H the vectors $(t_i^1, t_i^2, \dots, t_i^q)$ are used, where $1 \leq i \leq \alpha$
- Eqn. 1 is replaced by:

$$r^j = \bigoplus_{i=1}^{\alpha} t_i^j \quad \text{for each } j \quad (1 \leq j \leq q). \quad (2)$$

- The scheme is normally used to protect the authenticity, if you also want to protect the privacy you use a different privacy key which length is n bits.

In next section we will discuss the practical and theoretical consequences.

5. Practical and theoretical consequences

All schemes we discussed can be extended if we replace the modulo 2 sum by another modulo sum (e.g. modulo 53). We will wonder if the discussed schemes are useful. Consequences of the discussed schemes on the security of stream ciphers and DES will also be discussed.

5.1. Are previous schemes useful?

If you find the Vernam scheme impractical for your application, you find the discussed schemes also impractical. If however, you are dealing with national security (e.g., military and diplomacy) or you need unconditional security, the discussed schemes are interesting. If you use the Vernam one-time pad, you have to take into consideration that e.g., terrorists can modify your messages. As a consequence of terrorists attacks and of computer networks the problem of authenticity becomes more and more important, also in domains as the military or other governmental organizations. The discussed schemes allow to protect the authenticity unconditionally. The scheme discussed in Section 4.4 is preferable because the ciphertext expansion is about inexistent, while the length of the key is only about twice the length of the message. The security level obtained is less than the one which can be obtained ([2], [12], [22]), but the scheme is much more practical if long or very long messages are sent, while one can still choose the security level one wants. The key is used as in Vernam, so is random and distributed beforehand on a secure way. *Senders and receivers can easily handle messages with variable length.*

5.2. Stream ciphers protecting authenticity

Some authors, e.g. Denning [4] (pp. 144) say that stream ciphers have the disadvantages that the message can easily be modified. The schemes which we discussed here and certainly the one in Section 4.3 allow to modify stream ciphers such that they can be used to protect the authenticity. However their security is *no more unconditionally secure*, because stream ciphers generate pseudorandom, and their security is based on computationally complexity. If one adapts stream ciphers to protect authenticity, we suggest

to use a different key for the pseudorandom generator which will be used to protect the privacy and the one which will be used to protect the authenticity.

5.3. Hashing and unconditionally secure authentication

One could remark that the final solutions (proposed in Section 4.3 and in Section 4.4) hide the use of hashing, which seems the natural solution. However if hashing is used in these schemes, one loses the unconditional security. Indeed the difficulty to find two different texts which produce the same authenticator, is then based on the computational complexity. The solution of hashing can be used when unconditional security is not necessary, e.g. in the scheme discussed in Section 5.2.

Most of the schemes which we will discuss further on, do not protect the authenticity unconditionally, however some remarks are also valid for them.

5.4. The protection of privacy and authenticity together

In the schemes we have discussed we used a different key to protect the privacy from the one used to protect the authenticity. We suggest that a similar strategy would be used for all cryptosystems. Jueneman et. al. [15] suggested the same in their paper. Another example of the importance to use different keys will be discussed in Section 5.5.

We can also conclude that in a conventional system the protection of privacy and authenticity are *partly* (see the previous last paragraph of Section 4.3 and Section 6.1) separable, and that the use of a mode as e.g. CBC does not necessarily guarantee the protection of the authenticity. So we do not agree with the remark of Denning [4] (pp. 10), cited in the introduction (Section 1).

5.5. The consequences on the use of DES

Today DES [10] is probably the most used commercial crypto algorithm. An authentication scheme was proposed by the NBS [11]. We will show that if you protect both privacy and authenticity with the same key, that a fraudulent message may be easily injected, and that one can easily modify messages. Jueneman et. al. [15] suggested to use different keys to protect the authenticity and the privacy. Several attacks were presented in the case that the same key would be used, even if the NBS authentication method is used. They were able to modify the message without affecting the authenticator, however the received plaintext will (in almost all cases) be "garbage". The attack which will be presented now, allows an active eavesdropper to modify a message in a fraudulent one, he chooses! So in bank applications he is able to transfer money on his account such that the fraud will not be detected by the authentication system.

The attack presented here is an adaptation of an idea originating from Cloetens [3]. In [13] a realistic exhaustive keysearch machine was presented which would break DES in about four weeks, and would cost about \$1,000,000. The idea is to use such a machine. Hereto let us make some reasonable assumptions: the key is only modified once each four weeks, the privacy protection uses the same key as the authentication process and the active eavesdropper uses a known plaintext attack. He can then exhaustively determine the key, starting from a block of the ciphertext and a block of the plaintext. This attack is not influenced if the encryption system uses a mode. Once that key is found, the active eavesdropper can inject or modify messages. One could argue that by modifying the key

frequently enough, the attack is not more valid. However, it can still be used! Suppose that the sender and receiver modify their key each s seconds. The active eavesdropper can now stop his exhaustive keysearch machine each s seconds and try to find the next key. If the machine does this process enough randomly, it will *not* find a key after four weeks with a probability:

$$\left(\frac{x-1}{x}\right)^x \quad \text{where} \quad x = \frac{3600 \cdot 24 \cdot 7 \cdot 4}{s}.$$

In limit a key will be found after four weeks with a probability $1 - e^{-1}$, in eight weeks with a probability $1 - e^{-2}$, and so on. Once a key is found the active eavesdropper modifies the message as he wants.

Remark that the above attack is valid for all modes as long as the key, used to protect the privacy, is the same as the key, used to protect the authenticity, even if that key is modified frequently! Also, for several non-standard implementations of the DES such an attack is possible. Remark that the attack can not be avoided if for each message a different key is used (e.g. the first message is encrypted using key K_1 , the second with K_2 and so on). Indeed because the attack is even in limit ($x \rightarrow 0$) still valid. To realize the attack, it is enough to add a delay in the transmission and to have a described exhaustive machine which can be easily restarted. *Even in the case the key used to protect the authenticity is different from the one used to protect the privacy, care is necessary. Indeed if short messages are sent, it is trivial to prove that a similar attack is still valid. The time needed to break, increases only linearly with the length of the message.* This is a consequence of the linearly increasing time to calculate the authenticator, and as a consequence of the exhaustive attack. Similar as in the above case, it does not help to modify the key frequently. *Such situations of short messages can be forced with chosen text attacks!* Better exhaustive machines (than the one discussed in [13]) can make the discussed attacks cheaper, faster and so on. This discussion is certainly outside the scope of this paper (for more details see [7]).

Each encryption algorithm which is "similar" as DES suffers from this attack. The meaning of "similar" is explained in [18]. One could wonder if it would not be better to use always the so called "triple encryption" in order to avoid such and similar attacks. But even in that case we recommend that the key used to protect the privacy is *different* from the key used to protect the authenticity.

6. Can a public key scheme protects the authenticity without privacy?

6.1. Introduction

It is evident that the RSA scheme [19],[20] can protect (today) the authenticity of short messages (taken into consideration that a secure key is chosen [4]). However not so much research is done to protect the authenticity of long messages with RSA. Indeed, if one divides the message up in blocks and authenticates the blocks separately then an active eavesdropper can mix the blocks up, repeat them, delete some, and so on. To protect the authenticity of long messages, some authors propose the use of hashing functions, or propose to use DES and to distribute the key with RSA, or to use a protocol that

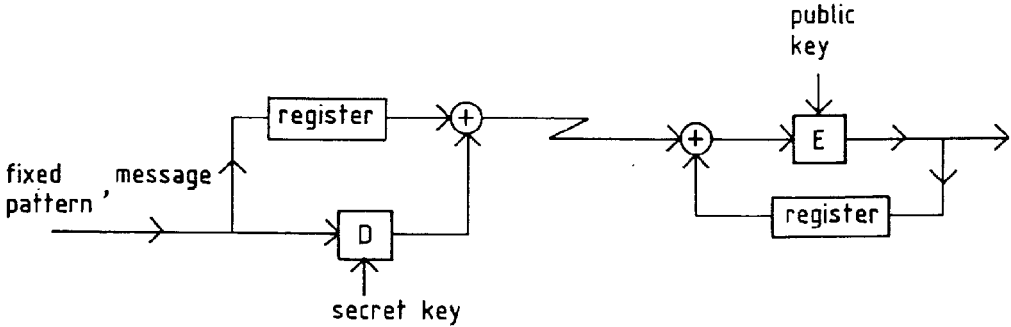


Figure 2: CBC mode with a public key algorithm to protect authenticity

“ping-pongs” the message from sender to receiver and back and so on. However these ideas have several disadvantages:

- hashing functions suffer mostly from “meet-in-the-middle” attacks [1]
- protocols, hashing functions and DES are extra costs
- ping-pong protocols slow down the communication
- hashing functions and DES do not allow the protection of the authenticity *without excluding the possibility to transmit secret information*. In some cases this is not acceptable e.g., as in arm limitation control [22] or if some country does not allow that encrypted messages are sent to foreign countries. Using a hashing function or DES, the authenticator can be replaced (e.g. partly) by secret information.
- random can not be used, because it can be misused for sending secret information.

We wonder if in a public key system *privacy and authenticity are completely separable* under the conditions mentioned (we don’t use hashing functions, or a conventional cryptosystem, or a ping-pong protocol, or random). We will now come up with a mode to protect the authenticity of long messages, however the presented scheme is not secure.

6.2. An insecure proposal

In the scheme we use a CBC mode (see Figure 2) to protect the authenticity. The sender uses a feedforward and the decryption algorithm with his secret key. The message is followed by a fixed pattern A as authenticator (a variable one could contain secret information, what we do not want). The receiver uses a feedback and the encryption algorithm with the public key of the sender. The initial contents of the registers used in the feedforward and feedback is fixed and publicly known, otherwise we protect partly the privacy. We call this initial contents I . Because a feedback has a large error propagation one could expect that this system is secure. However this scheme is insecure if an active eavesdropper know one block, e.g. I . Because the receiver uses the public key, an active eavesdropper is able to follow exactly what the contents of the register is in the device of

the receiver, he is also able to see what the output is and so on. He can now attack the protection by modifying arbitrary all sent blocks, except the last one corresponding with the authenticator. He will also modify the last transmitted block, however he calculates the modification such that the receiver still receives the authenticator A . Because he is able to do all the calculations the receiver does, he knows the previous last received message block M'_n . If the active eavesdropper would not have modified the transmitted blocks the receiver would have received M_n instead of M'_n . The last block transmitted by the sender is $M_n \oplus D(A)$, where $D(\cdot)$ is the decryption operation. If the active eavesdropper exors $M_n \oplus M'_n$ with the last block, the receiver will accept the message.

Using this attack the received message will probably be "garbage", nevertheless it will be accepted in an automatic system. For terrorists it does not matter if the received text is garbage, sabotage is enough. The active eavesdropper knows however the message that the receiver will receive and can try to come up with better "garbage".

The mode here proposed is insecure, and one can wonder if a secure mode exists. As long as no secure mode is found to protect the authenticity of *long* messages without protecting privacy and which satisfies the mentioned conditions, we have to conclude that authenticity can not be *completely* separated from privacy. This conclusion would be strange!

7. Conclusions

7.1. Overview of the presented results

We came up with several unconditionally secure authentication schemes. Nevertheless that they are not perfect in the sense of Simmons definition, the last unconditionally secure scheme proposed in our paper is more practical than the perfect ones.

We came up with stream ciphers which protect the authenticity.

We demonstrated that the ideas of Denning [4] about conventional systems are oversimplified. There exist conventional systems that protect the privacy but not the authenticity (e.g. Vernam one-time pad).

The protection of privacy and the protection of authenticity (and integrity) are partly separable, we wonder if they are completely separable.

7.2. Advices for users

If you need to protect privacy and authenticity use different keys for the different purposes.

Use triple encryption in DES. A standard (e.g. ANSI, ISO) which does not always use triple encryption is unacceptable. This is true as well as for the protection of the authenticity as well as for the protection of privacy (A full discussion would be too long and out of the scope of this paper, see [7] and [13]).

7.3. Acknowledgements

The author wants to thank Ernest Brickell for discussions about perfect authentication systems. These discussions were impossible without the visiting position at the

Univeristy of New Mexico, to who I am very grateful. The author thanks Henri Cloetens for his personal communication about the authentication with DES.

REFERENCES

- [1] S. G. Akl, "On the security of compressed encodings," *Advances in Cryptology, Proc. Crypto 83*, Santa Barbara, California, U. S. A, August 21 - 24, 1983, pp. 209 - 230.
- [2] E. F. Brickell, "A few results in message authentication," *Congressus Numerantium*, vol. 43, December 1984, pp. 141 - 154.
- [3] H. Cloetens, personal communication.
- [4] D. E. R. Denning, "Cryptography and Data Security", Addison - Wesley, Reading, Mass. , 1982.
- [5] Y. Desmedt, J. Vandewalle and R. Govaerts, "The mathematical relation between the economic, cryptographic and information theoretical aspects of authentication," *IEEE Intern. Symp. Inform. Theory*, St. Jovite, Quebec, Canada, September 26 - 30, 1983, Abstract of papers, pp. 93.
- [6] Y. Desmedt, "Analysis of the Security and New Algorithms for Modern Industrial Cryptography", Doctoral Dissertation, Katholieke Universiteit Leuven, Belgium, October 1984.
- [7] Y. Desmedt, F. Hoornaert and J.-J. Quisquater, paper in preparation.
- [8] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, no. 6, pp. 644 - 654, November 1976.
- [9] H. Feistel, "Cryptography and computer privacy," *Scientific American*, vol. 288, no. 5, May 1973, pp. 15 - 23.
- [10] FIPS publication 46 "Data Encryption Standard," *Federal Information Processing Standards Publ.*, National Bureau of Standards, January 1977.
- [11] FIPS publication 81, "DES modes of operation," *Federal Information Processing Standard*, National Bureau of Standards, U. S. Department of Commerce, Washington D. C. , U. S. A. , 1980.
- [12] E. N. Gilbert, F. J. MacWilliams, and N. J. A. Sloane, "Codes which detect deception," *Bell Syst. Tech. Journ.* , vol. 53, no. 3, March 1974, pp. 405 - 424.
- [13] F. Hoornaert, J. Goubert, and Y. Desmedt, "Efficient hardware implementations of the DES," *Advances in Cryptology, Proc. Crypto 84*, Santa Barbara, California, U. S. A, August 19 - 22, 1984 (Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1985), pp. 147- 173.
- [14] R. R. Jueneman, "Analysis of certain aspects of output feedback mode," *Advances in Cryptology, Proc. Crypto 82*, Santa Barbara, California, U. S. A, August 23 - 25, 1982, pp. 99 - 127.
- [15] R. R. Jueneman, S. M. Matyas and C. H. Meyer, "Authentication with manipulation detection code," *Proceedings of the 1983 IEEE Symposium on Security and Privacy*, Oakland, California, April, 1983, pp. 33 - 54.

- [16] D. Kahn, "Modern Cryptology," *Scientific American*, July 1966, pp. 38 - 46.
- [17] A. Konheim, "*Cryptography: A Primer*," John Wiley, Toronto, 1981.
- [18] J.-J. Quisquater, Y. Desmedt and M. Davio, "The importance of "good" key scheduling schemes (How to make a DES* scheme with ≤ 48 bit keys?)," presented at Crypto '85, Santa Barbara, August, 1985, to appear in: *Advances in Cryptology, Proc. Crypto 85*, (Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1986).
- [19] R. L. Rivest, A. Shamir, and L. Adleman, "On digital signatures and public-key cryptosystems," *Massachusetts Institute of Technology Technical Report LCS/TN-82*, Cambridge, Massachusetts, April 1977.
- [20] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, pp. 294 - 299, April 1978.
- [21] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell Syst. Tech. Journ.*, vol. 28, pp. 656 - 715, Oct. 1949.
- [22] G. Simmons, "Symmetric and Asymmetric Encryption," *ACM Computing Surveys*, vol. 11, no. 4, December 1979.
- [23] G. Simmons, "Authentication theory/coding theory," *Advances in Cryptology, Proc. Crypto 84*, Santa Barbara, California, U. S. A, August 19 - 22, 1984 (Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1985).
- [24] G. S. Vernam, "Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications," *Journal American Institute of Electrical Engineers*, v. XLV, pp. 109 - 115, 1926.

On the Security of Ping-Pong Protocols when Implemented using the RSA

(Extended Abstract)

Shimon Even¹ Oded Goldreich^{1,2} Adi Shamir³

ABSTRACT

The Security of the RSA implementation of ping-pong protocols is considered. It is shown that the obvious RSA properties, such as "multiplicativity", do not endanger the security of ping-pong protocols. Namely, if a ping-pong protocol is secure in general then its implementation using an "ideal RSA" is also secure.

1. INTRODUCTION

When studying the security of cryptographic protocols, one can take one of the following two approaches:

- 1) Distinguish between the security of the "high level structure" of the protocol and the security of the cryptosystems used for its implementation. The aim is, mainly, to better understand the structure of secure protocols and issues related to it. While studying the (security of the) structure of a protocol, it is assumed that the protocol is "implemented" with "ideal" cryptosystems. In other words, the cryptosystems are treated as if they were free of any properties which are not implied by the cancellation of encryption with the corresponding decryption. Such a treatment has usually an algebraic flavour. This approach can be found in [NS], [DY], [DLM], [DEK], [EG] and [EGL].
- 2) Study the security of a concrete implementation of the protocol with respect to the concrete cryptosystems used for the implementation. The aim is to develop concrete provably-secure protocols and to present a methodology for

¹ Computer Science Dept., Technion, Haifa, Israel.

² Currently in MIT, Lab. for Computer Science. Supported in part by a Weizmann Postdoctoral Fellowship.

³ Department of Applied Mathematics, Weizmann Institute, Rehovot, Israel.

developing and proving correctness of protocols. Characteristic tools in this approach are generalized notions of polynomial-time reductions.

This approach was pursued in [LMR], [GMR], [BGMR], [ACGM], [CF] and [GHY].

In this paper, we follow the first approach, but introduce some influences of the second approach. More specifically, we study the "high level structure" of protocols implemented using "ideal-RSA" cryptosystems (i.e. cryptosystems which possess only the obvious properties of the RSA). Our aim is to try to characterize the structure of protocols which are secure with respect to the obvious properties of the RSA. We restrict our study to a simple class of public-key protocols, known as ping-pong protocols. The reason for this restriction is that testing the security of protocols, from a slightly extended class, has been shown to be undecidable [EG]. We show that as far as the security of ping-pong protocols is concerned the obvious properties of the RSA do not give an adversary any additional edge. Put in other words, ping-pong protocols which are secure with respect to "ideal cryptosystems" - remain secure with respect to "ideal-RSA".

Our work was partially motivated by Denning's study of the weaknesses of the RSA implementation of a simple signing protocol [D]. We show that the weaknesses, pointed out in [Da] and [D], are due to the insecurity of the "high level structure" of the protocol and not to the fact that it was implemented using the RSA. We further discuss this issue in section 7.

2. PING-PONG PROTOCOLS AND THEIR SECURITY

In this section we recall the basic definitions regarding ping-pong protocols and their security problem.

2.1 Public Key Cryptosystems

Following [DH], a *public key cryptosystem (PKCS)* is a set of pairs of functions, such that every user X has an encryption function E_X and a decryption function D_X . Both functions are mappings from $\{0,1\}^*$ to $\{0,1\}^*$. There is a public directory containing all (X, E_X) pairs, while the decryption function D_X is known only to X . It is required that

- (1) For every $m \in \{0,1\}^*$, $E_X(D_X(m)) = D_X(E_X(m)) = m$.

D_X is the inverse function of E_X .

(2) It is infeasible to recover z when given $E_X(z)$ (and E_X).

For further details consult [DH] and [RSA].

In the rest of this paper we will refer to the encryption and decryption function as to operators. Operator words are defined (as usual) as the composition of operators; i.e. the operator word $\sigma_1 \cdots \sigma_2 \sigma_1$ maps $m \in \{0,1\}^*$ to $\sigma_1(\cdots \sigma_2(\sigma_1(m)) \cdots)$. Two operator words α and β are said to be equivalent if for every $m \in \{0,1\}^*$, $\alpha(m) = \beta(m)$. The equivalence between operator words will be denoted by \equiv .

Property (1) above implies the following

Operators' Cancellation Rules:

for every X , $E_X D_X \equiv D_X E_X \equiv$ the identity operator.

Property (2) above implies that user X can only apply encryption operators and his own decryption operator (i.e. operators from the set $\{D_X\} \cup \{E_Y: Y \text{ is any user in the network}\}$.)

2.2 Ping-Pong Protocols

Following [DY], a *ping-pong protocol* $P(X, Y)$ is a sequence $(\alpha_1, \alpha_2, \dots, \alpha_l)$ of operator words, such that $\alpha_{2i-1} \in \{D_X, E_X, E_Y\}$ and $\alpha_{2i} \in \{D_Y, E_X, E_Y\}$. Here X and Y are variables. In a concrete execution of the protocol they are substituted by the names of the participants.

An execution of protocol $P(\cdot, \cdot)$ by parties A and B , regarding the *initial message* $m_0 \in \{0,1\}$, proceeds as follows: In the first phase party A applies $\alpha_1[A, B]$ to the initial message m_0 , and transmits the result to B . In other words, in the first phase A transmits $m_1 = \alpha_1[A, B](m_0)$ to B . In the $2i$ -th phase B applies $\alpha_{2i}[A, B]$ to m_{2i-1} , and transmits the result ($m_{2i} = \alpha_{2i}[A, B](m_{2i-1})$) to A . In the $2i+1$ -st phase A applies $\alpha_{2i+1}[A, B]$ to m_{2i} , and transmits the result ($m_{2i+1} = \alpha_{2i+1}[A, B](m_{2i})$) to B . Here $\alpha_i[A, B]$ denotes the operator word which results from α_i by substituting $E_X [D_X]$ by $E_A [D_A]$ and $E_Y [D_Y]$ by $E_B [D_B]$.

2.3 Security of Ping-Pong Protocols

Following [DY], we say that a ping-pong protocol $P(\cdot, \cdot)$ is insecure if parties which did not take part in an execution of P (hereafter referred to as the *saboteurs*) can find out the initial message transmitted in that execution. To this end the saboteurs can initiate other executions of P and rely on the operators' cancellation rules (i.e. $E_X D_X \equiv D_X E_X \equiv$ the identity operator). It was shown [DEK] that it is sufficient to consider a single saboteur. A formal definition of insecurity follows:

Definition 1: Let $P(X, Y) = (\alpha_1, \alpha_2, \dots, \alpha_l)$ be a ping-pong protocol (as in the previous subsection).

Let A, B and S denote three distinct users.

Let $\Sigma_Z = \{D_Z\} \cup \{E_U: U \text{ is any user name}\}$. (Σ_X denotes the set of operators which may be applied by user X .)

Let $I(A, B, S) = \{\alpha_i[X, Y]: 1 \leq i \leq l \text{ and } X \neq Y \in \{A, B, S\}\}$. ($I(A, B, S)$ is the set of operator words which may be effected on messages in executions of the protocol $P(\cdot, \cdot)$ by two users out of A, B and S .)

Let $\Delta = (\Sigma_S \cup I(A, B, S))^*$.

The protocol $P(\cdot, \cdot)$ is *insecure* if there exists an operator word $\gamma \in \Delta$ such that $\gamma\alpha_1[A, B]$ is equivalent (under the operators' cancellation rules) to the identity operator. The operator word $\gamma\alpha_1[A, B]$ is called an *insecurity string*.

Remark 1: The above definition (in which only one saboteur is considered) is equivalent to a definition in which more than one saboteur is considered. The latter definition can be obtained by redefining Δ as follows

$$\Delta = ((\bigcup_{Z \neq A, B} \Sigma_Z) \cup (\bigcup_{X \neq Y \neq Z \neq X} I(X, Y, Z)))^*$$

A proof of the equivalence of the two definitions can be found in [DEK]. It is interesting to note that this equivalence does not hold for ping-pong protocols for more than two parties [EG].

Discussion: Note that under (the insecurity) Definition 1, the only properties of the public-key cryptosystem exploited by the saboteur (in his attack on the protocol) are the most obvious and general ones; namely the cancellation of encryption with the corresponding decryption. Definition 1 can be interpreted as considering only the security of the "high level structure" of the protocol.

Testing "high level security", may obviously provide evidence for the insecurity of a concrete implementation of the protocol, but it can not provide a proof that a particular implementation (with a particular public-key cryptosystem) is secure.

3. THE RSA AND ITS PROPERTIES

The RSA is the most popular implementation of the concept of a public-key cryptosystem. This system, presented in 1978 by Rivest, Shamir and Adleman [RSA] is widely believed to be secure. However, the encryption decryption functions of the RSA possess obvious properties which are not implied by the cancellation rules. We begin by presenting the RSA functions and continue by discussing their properties.

3.1 The RSA Functions

An instance of the RSA consists of a composite integer N which is the product of two large primes p and q , and two integers e and d such that $e \cdot d$ is congruent to 1 modulo $\phi(N)$ ($\phi(N) = (p-1)(q-1)$ is the Euler function).

To create an instance of the RSA, user A randomly picks two large primes p and q , and a number relatively prime to $(p-1)(q-1)$. User A computes $N = p \cdot q$ and $d = e^{-1} \bmod (p-1)(q-1)$. User A places $(A, (N, e))$ in the public directory and keeps all other information (in particular d, p and q) secret.

Encryption is done by raising the message to the e -th power modulo N ; while decryption is done by raising the message to the d -th power modulo N . Everyone can encrypt a message so that A can decrypt it. It is assumed that knowledge of the factorization of N is needed in order to be able to decrypt (and factorization is considered intractable). For simplicity let us identify the username (i.e. A) with the modulus N he uses. The encryption function of user N will be denoted by E_N and N 's decryption function will be denoted by D_N .

Formal Setting: Let us denote by Z_N the set of all residues modulo N (i.e. $Z_N = \{0, 1, 2, \dots, N-1\}$). By the above we have, for every $m \in Z_N$,

$$E_N(m) = m^e \bmod N \quad \text{and} \quad D_N(m) = m^d \bmod N$$

It can be easily shown that E_N and D_N cancel each other [RSA].

In addition to the cancellation of encryption by the corresponding decryption, the RSA possesses additional obvious relations - to be hereby discussed.

3.2 The Properties of the RSA

The main properties of the RSA are that the set of almost all its message space forms a group with respect to multiplication modulo N , and that the encryption and decryption operators are homomorphisms over this group. Note that the RSA induces a permutation over Z_N^* . For simplicity, we restrict the message space to $Z_N^* \subseteq Z_N$. This excludes only $p+q-1$ elements - a negligible fraction of the original message space (Z_N).

In subsection 3.3 it will be shown that all the other obvious properties of RSA can be derived from the above. This includes the fact that D_N is a homomorphism, the fact that $E_N(1)=1$ etc.

3.3 Axiomatization

In this subsection we present a complete axiomatization of the RSA properties discussed above. In the formal treatment, we will denote the message space by M_X . Recall that E_X and D_X are inverse permutations over M_X . A multiplication operator over M_X will be considered. It is axiomatized that this operator (denoted by μ_X) together with the set M_X forms an Abelian group. It is also axiomatized that E_X is a homomorphism of this group.

A0) *Cancellation Axiom*: For every $m \in M_X$ the following holds

$$D_X(E_X(m)) = E_X(D_X(m)) = m.$$

A1) *Abelian Group Axiom*: The set M_X and the binary operation μ_X form an Abelian group. That is, $\mu_X: M_X \times M_X \rightarrow M_X$ satisfies the followings (for every $m, m_1, m_2, m_3 \in M_X$):

$$A1.1) \mu_X(m_1, \mu_X(m_2, m_3)) = \mu_X(\mu_X(m_1, m_2), m_3)$$

$$A1.2) \mu_X(1, m) = \mu_X(m, 1) = m.$$

$$A1.3) \text{ There exists a } m^{-1} \in M_X, \mu_X(m, m^{-1}) = \mu_X(m^{-1}, m) = 1$$

$$A1.4) \mu_X(m_1, m_2) = \mu_X(m_2, m_1)$$

A2) *Homomorphism of the Encryption*: For every $m_1, m_2 \in M_X$ the following holds

$$E_X(\mu_X(m_1, m_2)) = \mu_X(E_X(m_1), E_X(m_2))$$

An equivalent formulation is achieved by generalizing the multiplication operator μ_X , to take arbitrary many arguments, and by introducing the multiplicative inverse function I_X .

The RSA Equalities

E0) *Cancellation of Encryption/Decryption*: For every $m \in M_X$, $D_X(E_X(m)) = m$ and $E_X(D_X(m)) = m$.

E1) *Nested Multiplication*: For every $m_1, m_2, \dots, m_d \in M_X$, and $0 < j - i < d$
 $\mu_X(m_1, \dots, m_i, \mu_X(m_{i+1}, \dots, m_j), m_{j+1}, \dots, m_d) = \mu_X(m_1, m_2, \dots, m_d).$

E2) *Redundant Multiplication*: For every $m \in M_X$, $\mu_X(m) = m$.

E3) *Redundant Identity*: For every $m_1, \dots, m_d \in M_X$, and $d \geq 1$
 $\mu_X(m_1, \dots, m_i, 1, m_{i+1}, \dots, m_d) = \mu_X(m_1, \dots, m_i, m_{i+1}, \dots, m_d)$

E4) *Inverse*: For every $m, m_1, \dots, m_d \in M_X$, $d \geq 0$ and $i \leq j$,

$$\mu_X(m_1, \dots, m_i, m, m_{i+1}, \dots, m_j, I_X(m), m_{j+1}, \dots, m_d) =$$

$$\mu_X(m_1, \dots, m_i, I_X(m), m_{i+1}, \dots, m_j, m, m_{j+1}, \dots, m_d) =$$

$$\mu_X(1, m_1, \dots, m_i, m_{i+1}, \dots, m_j, m_{j+1}, \dots, m_d)$$

E5) *Homomorphism of Inverse Operator*: For every $m_1, m_2, \dots, m_d \in M_X$, and $d \geq 2$, $I_X(\mu_X(m_1, m_2, \dots, m_d)) = \mu_X(I_X(m_1), I_X(m_2), \dots, I_X(m_d)).$

E6) *Cancellation of Double Inverse*: For every $m \in M_X$, $I_X(I_X(m)) = m$.

E7) *Generalized Homomorphism of Encryption/Decryption*:

For every $m_1, m_2, \dots, m_d \in M_X$ and $d \geq 2$ the following hold

$$E_X(\mu_X(m_1, m_2, \dots, m_d)) = \mu_X(E_X(m_1), E_X(m_2), \dots, E_X(m_d))$$

$$D_X(\mu_X(m_1, m_2, \dots, m_d)) = \mu_X(D_X(m_1), D_X(m_2), \dots, D_X(m_d))$$

E8) *Stability of Identity*: $E_X(1)=1$ $D_X(1)=1$ and $I_X(1)=1$.

E9) *Commutativity of Operators*: For every $m \in M_X$,

$$E_X(I_X(m))=I_X(E_X(m)) \text{ and } D_X(I_X(m))=I_X(D_X(m)).$$

4. SECURITY WITH RESPECT TO RSA PROPERTIES

In this section we define a new notion of insecurity: insecurity w.r.t RSA. Loosely speaking, a protocol is insecure w.r.t RSA if an adversary can seize the initial message by eavesdropping, initiating other executions of the protocol and taking advantage over the (10) equalities listed above.

In order to formally discuss the power of such an adversary, we have to study the algebra of expressions over the operator alphabet $\bigcup_X \{E_X, D_X, \mu_X, I_X\}$ w.r.t the equalities listed in Sec. 4.2. This algebra is best described by representing its expressions as rooted labelled trees and enforcing its equalities by tree manipulation rules.

4.1 The Algebra of Operator Trees

We start the description of the algebra by giving a representation of its expressions as rooted node-labelled trees.

Definition 2: An *operator tree* is recursively defined as follows:

A *constant* is an element of $\bigcup_X M_X$.

A *variable* may be assigned any element of $\bigcup_X M_X$.

An *atom* is a node labelled either a constant or a variable. An atom is an operator tree (rooted at the atom).

A *protocol node (P-node)* is a node labelled either E_X or D_X for some X . An operator tree rooted at a P-node v consists of the node v , an edge (v, u) and an operator tree rooted at u . The operator tree rooted at u is said to be a subtree hooked to v .

An *inverse-node (I-node)* is a node labelled I_X for some X . An operator tree rooted at an I-node v consists of the node v , an edge (v, u) and an operator tree rooted at u . (The operator tree rooted at u is said to be a subtree hooked to v .)

An *multiplication-node (μ -node)* is a node labelled μ_X for some X . An operator tree rooted at an μ -node v consists of the node v , a set of $d \geq 1$ edges $\{(v, u_i)\}_{i=1}^d$ and a set of operator trees rooted at u_1, u_2, \dots, u_d respectively. (The operator tree rooted at u_i is said to be a subtree hooked to v . Note that only a μ -node may have more than one son in an operator tree.)

As a first step towards defining the operator tree algebra we define two operator trees to be isomorphic if there is a "labelling and rooting preserving" isomorphism from one tree to the other. This isomorphism can be precisely defined as follows:

Definition 3: Two operator trees T_1 and T_2 are said to be *isomorphic* if one of the following hold:

- 1) Both trees are atoms, and either both are labelled by the same constant or both are labelled by the same variable.
- 2) For $i \in \{1, 2\}$, let T_i consist of the root v_i and d subtrees hooked to v_i denoted by $t_i^1, t_i^2, \dots, t_i^d$ respectively.

Then the labelling of v_1 and v_2 are equal and there exists a permutation π (over the set $\{1, 2, \dots, d\}$) such that for every $1 \leq j \leq d$, the subtree t_1^j is isomorphic to the subtree $t_2^{\pi(j)}$.

The equalities listed in Sec. 3.3 imply the following tree manipulation system. The system consists of 10 pairs of reduction rules, corresponding to these 10 equalities.

The Two-Way Reduction Rules: The notation $e_1(t) \rightarrow e_2(t) [e_1(t) <- e_2(t)]$ means that the subtree described by the expression $e_1(t) [e_2(t)]$ can be replaced by the subtree described by $e_2(t) [e_1(t)]$, in any operator tree. For every equality E_i ($0 \leq i \leq 9$) of the form $e_1(t) = e_2(t)$, we introduce a reduction rule (denoted R_i) $e_1(t) \rightarrow e_2(t)$ and a (reverse) reduction rule (denoted B_i) $e_1(t) <- e_2(t)$.

Finally, we define equivalence of operator trees as follows

Definition 4: Two operator trees are (*two-way*) *equivalent* if applying a sequence of reduction rules to one of them results in an operator tree which is isomorphic to the other. We stress that both the B_i and the R_i reduction rules may be used in this sequence of applications.

4.2 Properties of the R-Reductions Rules

The reduction rules discussed in subsection 4.1 consist of pairs R_i and B_i such that if R_i is $e_1(t) \rightarrow e_2(t)$ then B_i is $e_1(t) <- e_2(t)$. This system of reduction rules is clearly infinite, in the sense that one can apply an infinite sequence of reduction rules to every operator tree. We will consider the R_i ($0 \leq i \leq 9$) reduction rules hereafter called the *R-reduction rules*. The system of R-reduction rules is finite; that is, for every operator tree t there is a (finite) upper bound of the length of sequences of R-reduction rules which can be applied to t .

Lemma 3: Let n denote the number of nodes in the operator tree t . Then n^3 is an upper bound on the length of R-reduction sequences which can be applied to t .

Remark 2: The upper bound presented in Lemma 3 is tight up to a multiplicative constant. A demonstration of this fact is omitted from this extended abstract.

The finiteness of the R-reduction rules suggests the following

Definition 5: An operator tree is said to be *irreducible* if no R-reduction rule can be applied to it.

Corollary 1 (to Lemma 3):

For every operator tree t , there exists an operator tree r such that

- 1) r is an irreducible operator tree.
- 2) r is the result of applying a finite sequence of R-reduction rules to t .

Another appealing feature of the R-reduction rules is the insignificance of the order in which R-reduction rules are applied;

Lemma 4:

For every operator tree t there exists a **unique** operator tree r such that

- 1) r is an irreducible operator tree
- 2) r is the result of applying a finite sequence of R-reduction rules to t .

Lemma 4 can be proven by demonstrating⁴ that the R-reduction rules have the Church-Rosser Property [CR], and by using the Church-Rosser Theorem. By the Church-Rosser Theorem if a reduction system is finite (in the sense of Lemma 3) and has the Church-Rosser Property then each object has a unique irreducible object reachable from it. Lemma 4 suggests the following

Definition 7: The *reduced form* of t is an operator tree r such that r is irreducible and $t \rightarrow^* r$.

⁴ Such a demonstration is straightforward but tedious. An extensive study of general methods for proving Church-Rosser property of tree manipulation systems was conducted by Rosen [R].

Definition 8: Two operator trees are said to be *R-equivalent* if their corresponding reduced forms are isomorphic.

4.3 R-Reduction Rules versus Two-Way Reduction Rules

In this subsection we show that the R-reduction rules are as powerful as the two-way reduction rules. This is of much importance since, unlike the two-way reduction rules, the R-reduction rules can only be applied a finite number of times and yield a unique (irreducible) result.

Lemma 5: Two operator trees are R-equivalent if and only if they are two-way equivalent.

Corollary 4 (to Lemma 5): With respect to any operator trees t_1 and t_2 , the following are equivalent:

- 1) t_1 and t_2 can be proven to be equal in the proof system which consists of the axioms A0, A1 and A2 (of subsection 3.3) and the proof rule known as substitution.
- 2) The reduced form of t_1 is isomorphic to the reduced form of t_2 .

Thus, the R-equivalence "grasps" the structure of the algebra of operator trees.

4.4 The New Security Definition

Having presented the algebra of operator trees and its properties, we are ready to define insecurity with respect to this algebra. We will say that a protocol is insecure if a saboteur can construct an operator tree which is R-equivalent to the initial message sent from A to B . As in Definition 1 (subsection 2.3), the saboteur can apply any operator in his vocabulary as well as apply any instance of any protocol word to any message. Let us present a formal definition of this new insecurity notion.

Definition 9: Let $P(X, Y) = (\alpha_1, \alpha_2, \dots, \alpha_l)$ be a ping-pong protocol (as in subsection 2.2).

The protocol $P(\cdot, \cdot)$ is *RSA-insecure* if a saboteur S who does not know the initial message m_0 (in the execution of P by A and B) can construct an operator tree which is R-equivalent to m_0 . (This operator tree will be called the *insecurity tree* of protocol P .)

The trees that S can construct are recursively defined as follows:

- 1) S can construct the path $\alpha_1[A, B](m_0)$
- 2) S can construct an atom labelled by a constant.

- 3) Let t be an operator tree which can be constructed by S . Then S can construct the operator tree $D_Y(t)$ ($Y \neq A, B$ is any user in the net other than A or B) and the operator tree $E_X(t)$ (X is any user in the net).
- 4) Let t be an operator tree which can be constructed by S . Then S can construct the operator tree $\alpha_i[X, Y](t)$, where $1 \leq i \leq l$ and $X \neq Y$ are any two distinct users.
- 5) Let t_1, t_2, \dots, t_d be $d \geq 2$ operator trees which can be constructed by S . Then S can construct the operator tree $\mu_X(t_1, t_2, \dots, t_d)$ and the operator tree $I_X(t_1)$ (X is any user in the net).

Consider the following variant of the above definition:

Definition 10: The protocol $P(\cdot, \cdot)$ is *generically-insecure* if a saboteur S who does not know the initial message m_0 (in the execution of P by A and B) can construct an operator tree which is R-equivalent to m_0 , where the trees that S can construct are recursively defined by 1, 2, 3 and 4 above (without 5).

Remark 3: Note that the Definition 10 is identical to Definition 1 (the insecurity definition which appears in subsection 2.3).

(Note that if P is generically insecure then it has an insecurity tree which consists of a path with one atom (m_0) and all other nodes are P-nodes. Also recall Remark 1 in subsection 2.3.)

5. EQUIVALENCE OF THE TWO SECURITY DEFINITIONS

We are now ready to present the main result of this paper: the equivalence of the insecurity definitions presented in Definition 1 and Definition 9 (subsections 2.3 and 4.4) respectively. By Remark 3 (above) it suffices to show that

Main Theorem: A protocol P is RSA-insecure if and only if it is generically-insecure.

proof: Clearly, if the protocol P is generically-insecure then it is RSA-insecure. It is left to show that if P is RSA-insecure then it is generically-insecure.

Suppose that P is RSA-insecure, then it has an insecurity tree t which contains atoms, P-nodes and possibly I-nodes and μ -nodes. Consider an application of a R-reduction sequence to the operator tree t , resulting in the operator tree m_0 . Consider the node in t that was not reduced during this reduction process (it is

labelled by m_0); and the path in t from the root to this node. Denote the nodes on this path by n_0, n_1, n_2, \dots , and n_i . n_0 denotes t 's root and n_i the node labelled by m_0 which was not reduced in the reduction sequence.

First note that the path which results from n_0, n_1, \dots, n_i by omitting all μ -nodes and I-nodes can be constructed by the saboteur S . It is left to show that this path (the path which results from n_0, n_1, \dots, n_i by omitting all μ -nodes and I-nodes) is R-equivalent to m_0 . That is, that the P-nodes of this path can be paired in a non-interlacing manner such that the labels of the nodes of each pair are E_X and D_X (for some user X).

Throughout the R-reduction process, consider the path between the current root and n_i . Before the first reduction rule was applied this path consists of n_0, n_1, \dots, n_i . Consider the application of the i th R-reduction rule.

Case I: If the i th rule is not R0 then it does not cause the omission (or insertion) of any P-node in the path from the current root to n_i . Furthermore, it also does not change the order in which the P-nodes appear on the path from the current node to n_i .

[Note that we rely on the fact that n_i was not omitted during the R-reduction sequence.]

Case II: If the i th rule is R0, but it is not applied to a node on the path from the current root to n_i , then it does not effect the nodes on this path.

Case III: If the i th rule is R0 and it is applied to n_j which is on the path between the current root and n_i , then it causes the omission of n_j and some n_k . Note that both nodes are currently adjacent on the path between root and n_i . In this case we *pair* the node n_j with the node n_k .

Note that in the end of the reduction process the path between the current root and n_i consists of a single node: n_i . It is evident that we have paired all the P-nodes of the initial path between the root and n_i so that the pairs do not interlace and the labels of the P-nodes of each pair are E_X and D_X for some X .

We conclude by noting that concatenating the labels of the P-nodes on the initial path from root to n_i forms an insecurity string of the protocol P . Thus, P is generically-insecure.

QED

6. EXTENSIONS

The definitions and results of the previous sections can be extended in three directions:

- 1) The security of ping-pong protocols over extended operator-alphabet.
- 2) The security of multi-party ping-pong protocols.
- 3) Insecurity as the ability to accomplish the effect of a specified operator word.

In this extended abstract, we only deal with the last case. The first two cases will appear in the full version of this paper.

Insecurity as the Ability to Apply a Specific Operator Word

The insecurity definitions appearing in this paper can be rephrased as follows: Can a saboteur construct an operator tree which is equivalent, modulo a specific set of reduction rules, to the left inverse of the first word in the protocol (i.e. equivalent to $\alpha_1[A, B]^{-1}$)? [The reader is referred to subsection 4.4 for a definition of the set of operator trees which can be constructed by the saboteur.]

Following [E], we generalize the above notions of security, and consider the following question: Can a saboteur construct an operator tree which is equivalent, modulo a specific set of reduction rules, to a specific operator word β ? In case the saboteur can construct (using instances of the words of the protocol) such an operator word, we will say that the protocol is β -insecure, otherwise we say that the protocol is β -secure.

Using the appropriate sets of reduction rules, we derive definitions for β -RSA-insecurity and β -generic-insecurity. (In the first the set of reduction rules consists of all R reductions, while in the second the set of reduction rules consists only of RO.) The proof of the Main Theorem can be modified to yield the following:

Theorem 2: For every β , a protocol is β -RSA-insecure iff it is β -generically-insecure.

7. CONCLUDING REMARKS

In [D], Denning considered a signing protocol which consists of applying the user's decryption operator to the given document. Formally, this protocol consists of two phases: in the first one party S sends his counterparts (R) a document m (to be signed); in the second phase R applies D_R to m and replies with the result $D_R(m)$, which is considered to be R 's signature to m .

Using the fact that the message space of the RSA forms a group and that the encryption function is a homomorphism of this group, Denning demonstrated (by methods of Davida [Da] and others) that the RSA implementation of the above protocol is insecure. Namely, that S can get R 's signature to message m without R being willing to sign m .

The fact that the above protocol is D_R -RSA-insecure should be no surprise, since this protocol is obviously D_R -generically-insecure. In the above protocol, R does not have the choice of which messages he signs since he is assumed to play the protocol for any initial message. In order to get R 's signature to m all S needs to do is to send m to R in the first phase. Thus, the weaknesses pointed out by Denning *reflects only the weakness of the protocol she used, not a weakness of the RSA*. Furthermore,

Any successful attack on a ping-pong protocol which relies on the obvious properties of the RSA (listed in Sec. 2) can be transformed into a successful attack which works no matter which public-key cryptosystem is used to implement the protocol.

In other words, *generically-secure ping-pong protocols are immune against attacks which rely on the obvious properties of the RSA*.

ACKNOWLEDGEMENT

We wish to thank Silvio Micali for many intriguing discussions concerning the security of cryptographic protocols.

REFERENCES

- [ACGM] Awerbuch B., Chor B., Goldwasser S., and Micali S., "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*,
- [BGMR] Ben-Or, M., Goldreich, O., Micali, S., and Rivest, R.L., "A Fair Protocol for Signing Contracts", *Proc. of the 12th ICALP*, Lecture Note in Computer Science (194) Springer Verlag, 1985, pp. 43-52.
- [CR] Church, A., and Rosser, J.B., "Some Properties of Conversion", *Trans. Amer. Math. Soc.* 39, (1936), pp. 472-482.
- [CF] Cohen J.D., and Fischer, M.J., "A Robust and Verifiable Cryptographically Secure Election Scheme", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*,

- [Da] Davida G.I., "Chosen Signature Cryptanalysis of the RSA (MIT) Public Key Cryptosystem", Tech. Rep. TR-CS-82-2, Dept. of Electrical Engineering and Computer Science, Univ. of Wisconsin, Milwaukee, WI, Oct. 1982.
- [D] Denning D.E., "Digital Signatures with RSA and Other Public-Key Cryptosystems", *Comm. of the ACM*, Vol. 27, April 1984, pp. 388-392.
- [DLM] DeMillo, R., Lynch, N., and Merritt, M., "Cryptographic Protocols", *Proc. of the 14th ACM Symp. on Theory of Computation*, 1982, pp. 383-400.
- [DH] Diffie, W., and Hellman, M.E., "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.
- [DEK] Dolev, D., Even, S., and Karp, R.M., "On the Security of Ping-Pong Protocols", *Inform. and Control*, Vol. 55, 1982, pp. 57-68.
- [DY] Dolev, D., and Yao, A.C., "On the Security of Public-Key Protocols", *IEEE Trans. on Inform. Theory*, Vol. IT-29, 1983, pp. 198-208.
- [E] Even, S., "On the Complexity of Some Word Problems that Arise in Testing the Security of Protocols", presented in *NATO Advanced Research Workshop on Combinatorial Algorithms on Words*, Maratea, Italy, June 1984.
- [EG] Even, S., and Goldreich, O., "On the Security of Multi-Party Ping-Pong Protocols", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*, 1983, pp. 34-39.
- [EGL] Even, S., Goldreich, O., and Lempel, A., "A Randomized Protocol for Signing Contracts", *Comm. of the ACM*, Vol. 28, No. 6, pp. 637-647, 1985.
- [GHY] Galil Z., Haber S., and Yung M., "A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*,
- [GMR] Goldwasser, S., Micali, S., and Rackoff, C., "The Knowledge Complexity of Interactive Proof Systems", *Proc. of the 17th ACM Symp. on Theory of Computation*, 1985, pp. 291-304.
- [LMR] Luby, M., Micali, S., and Rackoff, C., "How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*, 1983, pp. 11-21.
- [NS] Needham, R.M., and Schroeder, M.D., "Using Encryption for Authentication in Large Networks of Computers", *Comm. of the ACM*, Vol. 21, No. 12, 1978, pp. 993-999.
- [RSA] Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Comm. of the ACM*, Vol. 21, February 1978, pp. 120-126.
- [R] Rosen, B.K., "Tree-Manipulation Systems and Church-Rosser Theorems", *Jour. of the ACM*, Vol. 20, No. 1, January 1973, pp. 160-187.

A Secure Poker Protocol that Minimizes the Effect of Player Coalitions

Claude Crêpeau

Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128 succursale "A", Montréal
Québec, Canada, H3C 3J7

1. Introduction

What can we expect from a poker protocol? How close to reality can we come?

From the outset of this research, we realized that a cryptographic protocol could achieve more security than its real life counterpart (with physical cards). But every protocol proposed until now was far from offering all the possibilities of a real deck of cards or could not achieve the full security we were expecting.

Since Shamir, Rivest and Adleman first stated a solution to the mental poker problem [SRA], many protocols trying to implement a fully secure game have been proposed. Although SRA proved in the two player case that such a solution is not possible from an information theoretic point of view, such solutions might be possible when the players' computational power is limited. The leakage of partial information, found by Lipton [Li], in the initial SRA protocol was fixed by Goldwasser & Micali [GM1], in the two player case, using probabilistic encryption. Unfortunately this scheme did not extend to a larger number of players. No complete solution to the multi-player version of the problem is yet known. All proposals make special assumptions, like the players' inability to establish secret communications [Yu]&[BF] or the existence of a trusted third party [FM].

To conceive a complete poker protocol, some constraints must be followed :

- Uniqueness of cards
- Uniform random distribution of cards
- Absence of trusted third party
- Cheating detection with a very high probability
- Complete confidentiality of cards
- Minimal effect of coalitions
- Complete confidentiality of strategy

Each card must appear once and only once, either in the deck or in the hand of one player. The only case when a card may appear more than once must be the result of some detectable cheating.

The hand of each player must depend on decisions made by every players, so that none of them has any control on his hand or on his opponents'. Every possible hand must have an equal probability and be accessible to all players.

No trusted party may be assumed, since any human can be bribed, and no machinery is entirely safe because no tamper proof device can be achieved.

Any attempt to cheat must be detected. The probability that a player may cheat without being detected must decrease very fast (exponentially) with respect to some security parameter that the players must decide before the game. Also the amount of work to accomplish the protocol should increase reasonably (polynomially) with respect to this parameter. The value of this parameter will depend on the confidence the players have towards each other, and the maximum computation power they can achieve.

No partial or total information about any card from the deck may be obtainable without the approval of every opponent. Also, no information may be obtained from a player's hand without his approval.

When more than two players are involved, some players could establish secret communication and exchange all their knowledge about the game, the protocol or any secret data involved in the protocol. Nonetheless they should not be able to take advantage of this. This information should be equivalent to the cards they separately have in their hands. In other words, as long as one player is not cheating, nobody can learn more about his hand, or about the cards in the deck, than what they can deduce from the cards in their coalition.

Finally, it is strategically very important in the game of poker that the losing players may keep their cards secret at the end of a hand. All the concept of bluffing is based on this fact. Therefore, an ideal protocol should neither force the players to reveal their hands nor any information leading to some knowledge about them.

Our protocol fully implements a secure game achieving the first six properties. Only the confidentiality of the strategy remains unsolved, as all players will be requested to reveal all their information in the detection of cheating phase of this protocol.

This protocol is based on the implementation of the new concept of Hiding-Revealing transactions. These transactions enable a party A to pick a value from a set known to a party B without letting B know which element A has picked.

Our multi-player protocol is a direct extension of our two player version. Let us first describe this simplified protocol. Suppose that P_1 and P_2 want to play a fair game of poker. Assume that each of them will cheat if he can do so (without being detected).

Assume a correspondance between the cards and the set $\{1,2,3,\dots,51,52\}$. So from now on, numbers from 1 to 52 will be used to describe the cards themselves.

2. Cards shuffling

To shuffle the deck of cards P_1 picks at random a permutation π_1 of $\{1,2,3,\dots,51,52\}$ and P_2 picks π_2 . Accessing any card will be done via these two permutations. Each player P_i locks his permutation π_i using a locking function that must satisfy some special properties described below (which two well known cryptosystems will be shown to satisfy). Then P_i posts the locked version of π_i . A card will be accessed by $\pi_2(\pi_1(k))$ for $k \in \{1,2,3,\dots,51,52\}$. Since a player doesn't know the permutation of his opponent, the value $\pi_2(\pi_1(k))$ cannot be predicted by either player. To get a card, a player

would like to compute $\pi_2(\pi_1(k))$ from k and the encoded permutations (with the collaboration of his opponent) without revealing the outcome of his calculation. This is clearly easy for P_2 , but tricky for P_1 .

3. Requirements

To play the game, each player will have to pick four functions, (L, U, H, R) according to the rules described below. Briefly, these functions will be used in the protocol to transfer information from one player to another. Suppose a player owns a set of values V . He will show a perfectly hidden version of his elements using the locking function L . The opponent will be able to select a value from V , without revealing which one, by choosing a locked element c , hiding it with H , asking the first player to use U in order to release some trap-door information about $H(c)$, and finally inverting the hiding he did previously, using R on $U(H(c))$ to read the value he has chosen. If H and R have some "good" properties, then the owner of the set V will not be able to determine the value chosen by his opponent, nor will his opponent learn anything on the other values in V .

The following properties summarize sufficient conditions to build our protocol. This general approach is followed by two fundamentally different implementations. The presentation is done in a general framework in order to avoid mixing practical implementation details with the high level aspects of the scheme.

Let V be the value space (in this case $\{1,2,3,...,51,52\} \subseteq V$).

Let S be the seed space.

Let C be the coded (locked) message space.

Let K be the unlocking key space.

Let M be the mask space.

Let A be the ambiguous value space.

Define

$L: V \times S \rightarrow C$	(locking function)
$U: A \rightarrow K$	(unlocking function)
$H: M \times C \rightarrow A$	(hiding function)
$R: M \times C \times K \rightarrow V$	(revealing function)

such that

(the full meaning of the following properties will become clear in the next sections)

- 1) $\forall v_1, v_2 \in V, \forall s_1, s_2 \in S, L(v_1, s_1) = L(v_2, s_2) \Rightarrow v_1 = v_2$
- 2) $\forall v \in V, \forall p, s \in S$, it is computationally infeasible to compute any information about v from $L(v, s)$.
- 3) $\forall v \in V, \forall p, s \in S$, it is computationally infeasible to compute s from v and $L(v, s)$.
- 4) $\forall m \in M, \forall v \in V, \forall s \in S, R(m, L(v, s), U(H(m, L(v, s)))) = v$

- 5) $\forall a \in A, \forall v_1, v_2 \in V, \forall s_1, s_2 \in S, \#\{m \in M \mid H(m, L(v_1, s_1)) = a\} = \#\{m \in M \mid H(m, L(v_2, s_2)) = a\}.$
- 6) $\forall v \in V, \forall p(m, s) \in M \times S$, it is computationally infeasible to compute m given $H(m, L(v, s))$ and $L(v, s).$
- 7) There exists some polynomial time algorithm for each of L, U, H, R , but U includes trap-door information which is computationally infeasible to derive from $L, H, R.$

Here $\forall_p x \in X$ means:

"for all $x \in X$, except a number of elements smaller than any polynomial fraction in $\log(\#X)$ ".

4. Protocol

Assuming the existence of such functions (two implementations are given later), we describe our protocol. Let $i \in \{1, 2\}.$

Preparation Protocol

- Each player P_i :
- Step 1. Defines $V_i, S_i, C_i, K_i, M_i, A_i.$
 - Step 2. Defines $L_i, U_i, H_i, R_i.$
 - Step 3. Picks π_i a permutation of $\{1, 2, \dots, 52\}.$
 - Step 4. Posts $L_i, H_i, R_i.$
 - Step 5. Picks $s_{i,1}, s_{i,2}, \dots, s_{i,52} \in S_i$ at random.
 - Step 6. Posts $l_i(v) = L_i(\pi_i(v), s_{i,v}), 1 \leq v \leq 52.$

From property 2, revealing $L_i(\pi_i(v), s_{i,v}), 1 \leq v \leq 52$, leaks no information on $\pi_i(v).$ But since only 52 values are possible for $\pi_i(v)$, it would be easy to determine $\pi_i(v)$ from $s_{i,v}$ if the latter were easily computable, by comparing $L_i(i, s_{i,v}), L_i(2, s_{i,v}), \dots, L_i(52, s_{i,v})$ with $l_i(v).$ This is why property 3 is essential.

Initially, every $v \in \{1, 2, 3, \dots, 52\}$ is marked as "free". Suppose P_2 wants to get a card,

Card Reading Protocol 1

- Step 1. P_2 picks at random a "free" value $v \in \{1, 2, 3, \dots, 52\}$ and marks v as "used".
- Step 2. P_2 asks P_1 for the value of $\pi_1(v).$
- Step 3. P_1 reveals $\pi_1(v)$ to $P_2.$
- Step 4. P_2 secretly computes $\pi_2(\pi_1(v))$, which is his card.

At the end of the game, P_1 will be able to produce a proof that the value revealed ($\pi_1(v)$) was correct by showing $s_{1,v}$ so that P_2 can compare $L_1(\pi_1(v), s_{1,v})$ with $l_1(v).$ P_1 cannot cheat on this, since $l_1(v)$ uniquely determines $\pi_1(v)$ by property 1.

At the end of the game, if P_2 claims to have the card $\pi_2(\pi_1(v))$ in his hand, he will be able to produce a proof of this by revealing $s_{2,\pi_1(v)}$ so that P_1 can then compare $L_2(\pi_2(\pi_1(v)), s_{2,\pi_1(v)})$ with $l_2(\pi_1(v))$. Again, P_2 cannot cheat on this, due to property 1. So this transaction is fully secured.

Now suppose P_1 wants to get a card,

Card Reading Protocol 2a (not sufficient in general)

- Step 1. P_1 picks at random a "free" $v \in \{1, 2, 3, \dots, 52\}$ and marks it as "used".
- Step 2. P_1 secretly computes $\pi_1(v)$.
- Step 3. P_1 picks $m \in M_2$ and computes $h = H_2(m, l_2(\pi_1(v)))$.
- Step 4. P_1 asks P_2 to compute the key k to h .
- Step 5. P_2 returns $k = U_2(h)$ to P_1 .
- Step 6. P_1 computes: $R_2(m, l_2(\pi_1(v)), k) = \pi_2(\pi_1(v))$ { by property 4 }.

Since H has property 5, P_2 gets strictly no information on $\pi_1(v)$ from h since any of the l_2 's may have been used with equal probability.

We call this (Step 3 to Step 6) a Hiding-Revealing transaction between P_1 and P_2 . In this transaction P_2 has revealed a secret value to P_1 , namely $\pi_2(\pi_1(v))$, without knowing which one he has given away. At the end of the game P_1 will be able to check this transaction when asking P_2 to reveal $s_{2,\pi_1(v)}$, P_1 can compare $L_2(\pi_2(\pi_1(v)), s_{2,\pi_1(v)})$ with $l_2(\pi_1(v))$. The only remaining problem is proving to P_2 that P_1 did not fool him in making him decode something else than $H_2(m, l_2(\pi_1(v)))$. How can we force P_1 to respect the protocol?

First, let us see how P_1 could cheat. Suppose P_1 asks P_2 to decode $h' = H_2(m, l_2(\pi_1(v')))$ for $v' \neq v$, instead of h . Then P_1 will get $\pi_2(\pi_1(v'))$ when claiming he has been accessing $\pi_2(\pi_1(v))$. This could be interesting for P_1 , for instance, if v' is marked as "used" because $\pi_2(\pi_1(v'))$ is in fact a card in P_2 's hand. This would allow P_1 to know one card of his opponent, at cost of not knowing one of his own. But at the end of the game, P_1 will not know what $\pi_2(\pi_1(v))$ is. So if P_1 is asked to reveal all the cards he should have accessed, he won't be able to do so. (In fact P_1 may decide to access the card later in the game but in that case the problem will carry over to this new card he pretends to read).

But P_1 could be more subtle than that. He could try to get partial information on many cards at once. Maybe P_1 claims to follow the protocol, when in fact he is asking P_2 to decode some special value $g = G(l_2(1), \dots, l_2(52))$ instead of h , hoping that $G'(l_2(1), \dots, l_2(52), U_2(g))$ returns relevant information (partial or total) on more than one entry of π_2 , for some easily computable functions $G: C^{52} \rightarrow A$, $G': C^{52} \times K \rightarrow V^{52}$ he has discovered. This way, he might find out what $\pi_2(\pi_1(v))$ is and get additional knowledge on some other cards.

Our general solution to this problem takes advantage of property 6. P_2 will ask P_1 , at the end of the game, to reveal $\pi_1(v)$, $s_{1,v}$ and m . This allows P_2 to compare $l_1(v)$ with $L_1(\pi_1(v), s_{1,v})$ and h with $H_2(m, l_2(\pi_1(v)))$. But this is not yet a proof of P_1 's fairness. Maybe P_1 computed m after he received

the answer k in Step 5.

Suppose P_1 uses g as defined earlier. After he gets $k=U_2(g)$ from P_2 , maybe he can deduce $\pi_2(\pi_1(v))$, additional information on π_2 , and some m such that $g=H_2(m, l_2(\pi_1(v)))$. However, if we force P_1 to publish a coded copy of m before making Step 4 of the transaction, then to prove a fair access to $\pi_2(\pi_1(v))$, P_1 would have had to compute m before learning k . But this is not possible from property 6 because from g and $l_2(\pi_1(v))$, P_1 cannot compute m . So P_1 is fair if and only if he knew m before the value of k was revealed to him.

To implement this solution, we use a (possibly trap-door), public one-way function O_1 that hides all partial information on its inputs and add one step to our protocol (notice that this modification is not necessary for one of the implementations proposed in section 5) :

Card Reading Protocol 2b

- Step 1. P_1 picks at random a "free" $v \in \{1,2,3,\dots,52\}$ and marks it as "used".
- Step 2. P_1 secretly computes $\pi_1(v)$.
- Step 3. P_1 picks $m \in M_2$ and computes $h=H_2(m, l_2(\pi_1(v)))$.
- Step 4. P_1 asks P_2 to compute the key k to h .
- Step 4a. P_1 posts $O_1(m)$.
- Step 5. P_2 returns $k=U_2(h)$ to P_1 .
- Step 6. P_1 computes: $R_2(m, l_2(\pi_1(v)), k) = \pi_2(\pi_1(v))$.

This way, P_2 can check later that P_1 knew m before Step 4 of the transaction. The fact that the one-way function hides all partial information is important. In some implementations it is possible for P_2 to compute $\{m | \exists i, 1 \leq i \leq 52 \text{ such that } H_2(m, l_2(i))=h\}$. If O_1 did leak some information, then the correct m could be found or the set of possible candidates could be reduced, according to the leaked information, and P_2 would gain some knowledge about $\pi_1(v)$.

During the game, cards can be picked from the deck or discarded just by marking (with "used" or "discarded", respectively) the appropriate element in $\{1,2,3,\dots,52\}$. (e.g. to discard $\pi_2(\pi_1(v))$ just mark v as "discarded".)

At the end of the game all secret information must be published as proofs of fairness of the players. But some care must be taken in the implementations where the O_i functions are not used, to avoid revealing some secret item too soon. Otherwise a cheating opponent could forge a fairness proof from what he just learned. So, proofs of fairness must be done in the following way. To prevent a cheater from forging a proof, each player must execute Step 1 before anyone does Step 2 since Step 2 is used to prove that the values revealed in Step 1 were the correct ones. Also each one must execute Step 2 before anyone does Step 3 since learning the s 's may be a clue to the successful forging of m .

Proof of Fairness Protocol

Each player P_i reveals:

Step 1. $\pi_1(v), \pi_2(\pi_1(v))$ for each v he has accessed.

Step 2. All $m \in M$ used for some Hiding-Revealing operations.

Step 3. $\pi_i(v)$ and $s_{i,v}$, $1 \leq v \leq 52$.

This enables the opponent to check the transactions and to be sure that no cheating took place. The generalization to more than two players is found in section 6.

5. Implementations

We now propose two implementations of this protocol. The first is based on RSA [RSA] and the efficient probabilistic encryption scheme of Blum & Goldwasser (BG) [BG]. The second is based on the probabilistic encryption scheme of Goldwasser & Micali (GM) [GM]. This first version matches the general pattern given above.

5.1. Using RSA/BG.

Let P be one of the players. P selects p, q two large primes; large enough for the least 6 RSA bits to be $1/2 + (1/\text{poly}(\log(pq)))$ -secure (see [CG]). Let $n=pq$. P selects e, δ such that $e\delta \equiv 1 \pmod{\phi(n)}$. Then define $V=\{0,1,2,3,\dots,63\}$, $S=K=M=A=\mathbb{Z}_n^*$, $C=V \times S$. $(x \downarrow n)$ denotes the least n significant bits of x and \oplus denotes the bit-by-bit exclusive-OR.

Functions

$$\begin{aligned} L(v,s) &= ((s \downarrow 6) \oplus v, (s^e \bmod n)) \\ U(c) &= (s(c)^\delta \bmod n) \\ H(m,c) &= (m^e s(c) \bmod n) \\ R(m,c,k) &= (((m^{-1}k) \bmod n) \downarrow 6) \oplus v(c) \\ O(m) &= BG(m,s) \end{aligned}$$

Where m^{-1} is the multiplicative inverse of $m \pmod{n}$ and $c=(v(c), s(c))$. BG is the Blum-Goldwasser encryption function (see [BG]); in this implementation, one can use BG in the following way. P picks a random $s_0 \in S$, and compute $s_k = s_{k-1}^e \bmod n$, $1 \leq k \leq \lfloor n/6 \rfloor$. P then posts $s_{\lfloor n/6 \rfloor}$ and $\langle (s_0 \downarrow 6)(s_1 \downarrow 6) \dots (s_{\lfloor n/6 \rfloor - 1} \downarrow 6) \rangle \oplus m$. Here $\langle \dots \rangle$ denotes the concatenation of the given blocks of 6 bits. According to [BG], no partial information about m can efficiently be computed from these two public values. At the end of the game P will have to reveal s , then everybody will be able to compute s_1, s_2, \dots and recover m by inverting the \oplus . It will also be possible to check that this is the correct s_0 since $s_{\lfloor n/6 \rfloor}$ is uniquely decodable.

Theorem: L, U, H, R satisfy properties 1 to 7 (assuming inverting RSA is hard).

Proof:

$$\begin{aligned}
 1) & \forall v_1, v_2 \in V, \forall s_1, s_2 \in S, L(v_1, s_1) = L(v_2, s_2) \\
 & \Rightarrow ((s_1 \downarrow 6) \oplus v_1, (s_1^e \bmod n)) = ((s_2 \downarrow 6) \oplus v_2, (s_2^e \bmod n)) \\
 & \Rightarrow (s_1^e \bmod n) = (s_2^e \bmod n) \text{ and } (s_1 \downarrow 6) \oplus v_1 = (s_2 \downarrow 6) \oplus v_2 \\
 & \Rightarrow s_1 = s_2 \text{ and } (s_1 \downarrow 6) \oplus v_1 = (s_2 \downarrow 6) \oplus v_2 \\
 & \Rightarrow v_1 = v_2
 \end{aligned}$$

2) Since the 6 least significant bits of s are $1/2 + (1/\text{poly}(\log(n)))$ -secure.

3) RSA is assumed hard to invert. Being given the last 6 bits of s can at best speed up finding s by a factor of 64.

$$\begin{aligned}
 4) & \forall m \in M, \forall v \in V, \forall s \in S, H(m, L(v, s)) = ((m^e s^e) \bmod n) = ((ms)^e \bmod n) \\
 & \Rightarrow U(H(m, L(v, s))) = (ms) \bmod n \\
 & \Rightarrow R(m, L(v, s), U(H(m, L(v, s)))) = ((m^{-1}(ms \bmod n) \bmod n) \downarrow 6) \oplus (s \downarrow 6) \oplus v \\
 & = ((s \bmod n) \downarrow 6) \oplus ((s \bmod n) \downarrow 6) \oplus v = v
 \end{aligned}$$

$$\begin{aligned}
 5) & \forall v \in V, \forall s \in S, \forall a \in A, \{m \in M \mid H(m, L(v, s)) = a\} = \{m \in M \mid (ms)^e \bmod n = a\} \\
 & = \{a^{\delta} s^{-1} \bmod n\} \\
 & \Rightarrow \#\{m \in M \mid H(m, L(v, s)) = a\} = 1
 \end{aligned}$$

6) Suppose we have $x \in \mathbb{Z}_n^*$. If a polynomial fraction in $\log(\#M \times C)$ values of $H(m, c)$ were easy to invert given c , then we could choose $a_k = s(c_k) r_k^e x$ for polynomially many random c_k, r_k and with a very high probability, find a solution to $a_{k_0} = H(m, c_{k_0})$ (k_0 is one of the values attempted), using that inversion algorithm. One can check that $m r_{k_0}^{-1}$ would then be a solution to $(m r_{k_0}^{-1})^e \equiv x \pmod{n}$. This would imply we can invert RSA.

7) Clearly L, U, H, R are easy to compute.

□

In this first implementation we need O to solve the cheating problem. But in this next one, no O is required. Data expansion may be greater in this second version but the simplicity obtained worth the difference.

5.2. Using Probabilistic Encryption (GM).

Let P be one of the players. P selects p, q two large primes. Let $n = pq$. P selects η such that $\left(\frac{\eta}{p}\right) = \left(\frac{\eta}{q}\right) = -1$. (where $\left(\frac{x}{p}\right)$ is the Legendre symbol of x over p). Then define $V = K = \{0, 1\}^6 = \{0, 1, 2, 3, \dots, 63\}$, $S = (\mathbb{Z}_n^*)^6$, $C = A = \mathbb{Z}_n^* [+1]^6$ and $M = V \times S$. (where $\mathbb{Z}_n^* [+1] = \{x \mid x \in \mathbb{Z}_n^* \text{ \& } \left(\frac{x}{n}\right) = +1\}$ and $\left(\frac{x}{n}\right)$ is the Jacobi symbol of x over n)

Denote $x=(x_1,x_2,x_3,x_4,x_5,x_6) \in X$, where (x,X) is any of $(v,V), (c,C), (s,S), (k,K)$ or (a,A) . Denote also $m=(v(m),s(m)) \in M=V \times S$ and $(c \otimes c')_j = (cc'_j \bmod n)$.

Functions

$$\begin{aligned} L(v,s) &= (\lambda(v_1,s_1), \lambda(v_2,s_2), \lambda(v_3,s_3), \lambda(v_4,s_4), \lambda(v_5,s_5), \lambda(v_6,s_6)) \\ U(a) &= (v(a_1), v(a_2), v(a_3), v(a_4), v(a_5), v(a_6)) \\ H(m,c) &= L(m) \otimes c \\ R(m,c,k) &= v(m) \oplus k \end{aligned}$$

Where $\lambda(x,y) = y^2 \eta^x \bmod n$ and $v(x) = \begin{cases} 0 & \text{if } x \text{ is a quadratic residue mod } n \\ 1 & \text{if } x \text{ is not a quadratic residue mod } n \end{cases}$. Note that these functions are inspired by those defined in [GM2].

Theorem: L, U, H, R satisfy properties 1 to 7. (assuming the quadratic residuosity assumption (QRA))

Proof:

$$1) \forall v, v' \in V, \forall s, s' \in S, L(v,s) = L(v',s')$$

$\Rightarrow v=v'$ since GM is uniquely decodable.

2) Known property under QRA. (see [GM2])

3) True under QRA. Because the ability to compute s from v and $L(v,s)$ would allow to extract square roots, hence factor n which is hard under QRA.

$$4) \forall v \in V, \forall s \in S, \forall m \in M, R(m, L(v,s), U(H(m, L(v,s))))$$

$$= v(m) \oplus U(H(m, L(v,s)))$$

$$= v(m) \oplus v(m) \oplus v \text{ since } v(s(m)_j^2 \eta^{v(m)_j} s_j^2 \eta^{v_j}) = v(m)_j \oplus v_j, 1 \leq j \leq 6$$

$$= v$$

$$5) \forall v \in V, \forall s \in S, \forall a \in A, \{m \in M \mid H(m, L(v,s)) = a\} = \{m \in M \mid L(m) \otimes L(v,s) = a\}$$

$$= \{m \in M \mid s(m)_j^2 \eta^{v(m)_j} \bmod n = a s_j^2 \eta^{-v_j} \bmod n, 1 \leq j \leq 6\}$$

$$= \{m \in M \mid s(m)_j^2 \eta^{v(m)_j} \bmod n = x_j \eta^{v_j} \bmod n, 1 \leq j \leq 6\} \text{ with } a s_j^2 \eta^{-v_j} = x_j \eta^{v_j} \text{ where } x_j \text{ is a quadratic residue mod } n.$$

$$= \{m \in M \mid s(m)_j^2 = x_j \text{ and } v(m)_j = v'_j, 1 \leq j \leq 6\}$$

But each of these 6 equations have 4 solutions.

$$\text{So } \#\{m \in M \mid H(m, L(v,s)) = a\} = 4^6.$$

6) Suppose we have $x \in \mathbb{Z}_n^* [+1]$. If a polynomial fraction in $\log(\#M \times C)$ values of $H(m, c)$ were easy to invert given c , then we could choose $a_{k,1} = c_{k,1} s_{k,1}^2 \eta^{v_{k,1}} x \bmod n$, with $a_{k,2}, a_{k,3}, a_{k,4}, a_{k,5}, a_{k,6}$ random numbers in $\mathbb{Z}_n^* [+1]$, for polynomially many random $c_{k,2}, \dots, v_{k,6}$ and with a very high probability, find a solution to $a_{k,0} = H(m, c_{k,0})$ (k_0 is one of the values

attempted), using that inversion algorithm. One can check that $ms_{k_0,1}^{-2}\eta^{-v_{k_0,1}}$ would then be a solution to $(s_{k_0,1}^{-1}s_1(m))^2\eta^{v_1(m)-v_{k_0,1}}\equiv x \pmod{n}$. This would imply we can decide quadratic residuosity \pmod{n} (in contradiction with QRA).

7) Clearly L, U, H, R are easy to compute.

□

This implementation is particular because no encoding of $m \in M$ is needed to prove that someone knew m before using it in the Hiding-Revealing transaction. This is because of the next result:

Theorem: m cannot be computed from $U(H(m, c))$ and c , for any $m \in M, c \in C$.

Proof: $U(H(m, c))$ is independent of $s(m)$.

Suppose P_j makes a hiding-revealing transaction with P_i . If he doesn't know m before the transaction he cannot know it after since $s(m)$ cannot be computed from $U(H(m, c))$. So to prove that he knew m before the transaction, P_j just has to prove he knows it at the end of the game, since nothing has revealed this value to him after the transaction or later in the game.

6. Multi-Player protocol

We now extend the previous protocol to the multi-player problem. Suppose that P_1, P_2, \dots, P_j want to play poker. The preparation protocol is identical to the two-player version. A card will be accessed as $\pi_j(\pi_{j-1}(\pi_{j-2}(\dots(\pi_2(\pi_1(v))))))$ for $v \in \{1, 2, 3, \dots, 52\}$. Suppose $P_n, n \in \{1, 2, \dots, j\}$ wants to get a card.

Card reading Protocol 3

- | |
|---|
| <p>Step 1. P_n picks at random $v \in \{1, 2, 3, \dots, 52\}$
and marks it as "used".</p> <p>Step 2. $v_1 = v$</p> <p>Step 3. FOR $i=1$ TO $n-1$ DO</p> <p>Step 3.1 P_n asks P_i to reveal $\pi_i(v_i)$;</p> <p>Step 3.2 P_i answers $\pi_i(v_i)$ publicly;</p> <p>Step 3.3 P_n sets $v_{i+1} = \pi_i(v_i)$;</p> <p>Step 4. P_n secretly computes $v_{n+1} = \pi_n(v_n)$;</p> <p>Step 5. FOR $i=n+1$ TO j DO</p> <p>Step 5.1 P_n secretly gets $v_{i+1} = \pi_i(v_i)$
using the Hiding-Revealing protocol with P_i;</p> <p>Step 6. P_n's new card is v_{j+1}</p> |
|---|

This way, P_n computes $v_{j+1} = \pi_j(\pi_{j-1}(\pi_{j-2}(\dots(\pi_2(\pi_1(v))))))$ and nobody except himself knows $v_{n+1}, \dots, v_j, v_{j+1}$. All the proofs of fairness described in the two-player version can be used again in the multi-player case. Again depending on the implementation, O_i 's may be needed to obtain proofs of fairness in the Hiding-Revealing protocol. Finally the order of revelations at the end of the game must be the same as in the protocol for two players when those O_i 's are not used.

7. Security against player coalitions

The main improvement of this protocol is protection against coalitions. If some players form a coalition, they will not get any advantage from it, other than learning each other's hand. Since every card is accessed through each permutation, no subset of the players can know anything about the cards of the other players, other than knowing that they are not the cards within the coalition. Assume some player P_i is not a member of a coalition (by this we mean that P_i does not reveal any private information to any other player), then by the construction of the protocol we know that the values $v_{i+1}, \dots, v_j, v_{j+1}$ are secret and known only to himself for each of his cards. Since the v_{j+1} 's actually identify his cards, nobody has any information on them (unless someone has not followed the protocol but in that case he will be detected at the end of the game). Similarly, no coalition can influence the cards drawn by an honest player.

8. Playing other games

Our new protocol can be extended to play almost any card game, as well as other kind of games, such as Scrabble. Fortune and Merrit [FM] mentioned games that could not be played with their protocol. With our protocol, one can play any game where cards have to be exchanged between players more than once, or where cards may be dealt and discarded many times. Suppose for instance that P_n wants to get a card from P_i 's hand.

Card exchange Protocol

- Step 1. P_i reveals, in a random order, a locked version of his cards
($L_i(x, s_x)$) for each card x in his hand, with a random seed $s_x \in S_i$).
- Step 2. P_n picks one of them, tells which one to P_i
- Step 3. P_i returns $L_n(x, s)$ with a random seed $s \in S_n$
where x is the card P_n wants.
- Step 4. P_n recovers x using H_n, U_n, R_n .

This way, only P_i and P_n know which card was exchanged. In fact, everybody will be able to check this operation at the end of the game when P_i reveals the s_x 's. Also no information, like the identity of the previous players who ever had the card, is embedded with it. One might think that the following solution is sufficient, but if fact it is not. P_n picks a value that P_i claims as "used" and reads it using the Hiding-Revealing protocol. P_n has to inform P_i of which card he has picked so, that P_i knows that this card is no longer in his hand. But if later P_i have to pick a card from P_n 's

hand, he will be able to choose the same card P_n had picked before since he knows how to access it.

One can notice a problem with discarding in our standard protocol if the game played allows several dealing and discarding of cards. Suppose cards were dealt and then some were discarded. Suppose we deal some more cards and then discard some of the cards in our new hands (there are variants of poker in which players may ask twice to change cards). On the second discarding operation each player knows if the discarded cards of his opponents, come from the initial hand or from the new one dealt, since the "discard" markers are tagged on to the public values $v \in \{1, 2, \dots, 52\}$. This information may be compromising. To solve this, a card should be discarded by revealing a coded version of it and declaring it discarded (This idea was introduced in [Yu]). If P_i wants to discard the card $\pi_j(\pi_{j-1}(\pi_{j-2}(\dots(\pi_2(\pi_1(v))\dots)))$, instead of marking v as "discarded", he does the following:

Discarding Protocol

P_i posts $L_i(v, s)$ for some $s \in S_i$ and declares it "discarded".

At the end of the game, when P_i reveals v and s , the other players will be able to determine which cards P_i has accessed, which are still in his hand and which were discarded during the game. Just like in the exchange of cards, this reveals no information on its origin.

Another interesting feature of this protocol is the ability to return cards into the deck. Initially, each player goes through the Preparation Protocol, exactly as before and uses the other protocols for the other standard operations. Suppose that some players (maybe only one) want to return some cards (maybe only one) into the deck for the n^{th} time. It would not suffice to change their marks from "used" to "free" because this would allow the next person who select one of these cards to know that it had been in someone's hand previously. The entire deck, including the cards just returned to it, must be re-shuffled by:

Card Returning Protocol(part 1)

Each player P_i :

- Step 1. Picks $\pi_{i,n}$ a new permutation of $\{1, 2, \dots, 52\}$.
- Step 2. Picks $s_{i,1,n}, s_{i,2,n}, \dots, s_{i,104,n} \in S_i$ at random.
- Step 3. Posts $L_{i,n}(v) = L_i(\pi_{i,n}(v), s_{i,v,n})$, $1 \leq v \leq 52$.
- Step 4. Posts $L_{i,n}(v+52) = L_i(\pi_{i,n}^{-1}(v), s_{i,v+52,n})$, $1 \leq v \leq 52$.
- Step 5. Sets $\pi_i = \pi_{i,n}$.

When this is all done, the players will have to read backwards the new origin of the cards they have under their control (in their hands and among those they have discarded). They will not be able to cheat on this since the other players will check the correspondance between the cards had under each of the $\pi_{i,n}$.

Card returning Protocol(part 2)

Each player P_i , for each card c under his control he wishes to keep,
 Step 1. sets $c_j = c$
 Step 2. FOR $l=j$ DOWNT0 $i+1$
 Step 2.1 Reads $\pi_{i,l,n}^{-1}(c_l)$ using the Hiding-Revealing protocol with P_l ;
 Step 2.2 Sets $c_{l-1} = \pi_{i,l,n}^{-1}(c_l)$;
 Step 3. Sets $c_{i-1} = \pi_{i,i,n}^{-1}(c_i)$;
 Step 4. FOR $l=i-1$ DOWNT0 1
 Step 4.1 Reads $\pi_{i,l,n}^{-1}(c_l)$ using the Hiding-Revealing protocol with P_l ;
 Step 4.2 Sets $c_{l-1} = \pi_{i,l,n}^{-1}(c_l)$;
 Step 5. Declares c_0 (the origin of c) as "used".

These operations can all be verified when the $\pi_{i,l,n}$'s and the $s_{i,l,n}$'s, for $1 \leq l \leq 104$, are revealed. Notice that this feature enables the implementation of "a Scrabble Protocol that minimizes the effect of player coalitions". To do this, change cards into letters and the deck into the box of letters. Then the dealing of letters is similar to the dealing of cards and so on... But since letters can be returned into the box, this last feature is necessary to implement that game.

9. Open Problem

Nothing is quite perfect. There is one thing our protocol cannot do. The strategy of each player is completely revealed at the end of each game since our protocol asks everyone to show every information involved in the protocol. It makes it impossible for the players to bluff. Real poker players would never accept to play such a game. Although we believe such a protocol can be achieved, we do not have a complete solution yet.

10. Acknowledgements

I wish to thank Gilles Brassard, Pierre McKenzie and Jean-Marc Robert for fruitful discussions and for the numerous comments they made on the protocols.

11. REFERENCES

- [BF] Banary, I. and Furedi, Z. "Mental Poker with Three or More Players", in *Information and Control*, 59 (1983), pp. 84-93.
- [BG] Blum, M. and Goldwasser, S. "An Efficient Probabilistic Public-Key Encryption Scheme which Hides All Partial Information", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp.289-299.
- [CG] Chor, B. and Goldreich, O., "RSA/Rabin Least Significant Bits Are $1/2+1/\text{poly}(\log n)$ Secure", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp.303-313.

- [FM] Fortune, S. and Merrit, M., "Poker Protocols", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp.454-464.
- [GM1] Goldwasser, S. and Micali S., "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information", in *Proceedings of the 14th Annual ACM symp. on Theory of computing*, ACM-SIGACT, May 1982, pp. 365-377.
- [GM2] Goldwasser, S. and Micali S., "Probabilistic Encryption", in *J. Comput. System Sci.*, 28 (1984), pp. 270-299.
- [RSA] Rivest, R., Shamir, A. and Adleman L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", in *Communications of the ACM* 21,2 (February 1978), pp. 120-126.
- [SRA] Shamir, A., Rivest R. and Adleman L., "Mental Poker", MIT Technical Report, 1978.
- [Yu] Yung, M., "Cryptoprotocols: Subscription to a Public Key, The Secret Blocking and the Multi-Player Mental Poker Game", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp.439-453.

A Framework for the Study of Cryptographic Protocols

Richard Berger ()*

*Sampath Kannan (**)*

*René Peralta (***)*

Computer Science Division

University of California

Berkeley, California.

ABSTRACT

We develop a simple model of computation under which to study the meaning of cryptographic protocol and security. We define a protocol as a mathematical object and security as a possible property of this object. Having formalized the concept of a secure protocol we study its general properties. We back up our contention that the model is reasonable by solving some well known cryptography problems within the framework of the model.

1. Introduction.

It can be argued that cryptographers have been able to provide satisfactory solutions to only the simplest among the problems involving transactions between mutually suspicious parties. In this category lie problems like flipping coins [1], exchange of a single bit [2] (or a fraction of a bit [3]), demonstrating the truth of some boolean predicates on the secret keys [4], and the Oblivious Transfer [5] [6]. Harder problems

(*) Research sponsored in part by GTE fellowship. (**) Research sponsored by the Helen and George Pardee Fellowship (*** Research sponsored in part by NSF grant MCS-82-04506 and by Universidad Católica de Chile.

which in our opinion have not been completely solved include exchange of secret keys [7], certified mail [8], contract signing [9], and mental poker [10] [11]. The published solutions to the latter problems either have not been proven secure or use the cryptographic definition of one-way function. Cryptographers use the term one-way function to mean a function which has whatever it takes to make its use in cryptographic protocols secure. In particular, (cryptographic) one-way functions reveal no partial information about their inverse value. Even though one-way functions are useful theoretical objects, the actual encryption functions available in the literature are not one-way functions in this strong sense. Even the probabilistic encryption methods of Blum, Blum, and Shub [12] [13] have not been shown secure under multiple encryptions of the same message or of functionally related messages. It is not clear that there exists secure solutions for the harder problems mentioned above which assume only the hardness of inverting an encryption function. If these solutions do exist, it is likely that considerably more powerful theoretical tools will have to be developed before they can be found. The development of such tools is the objective of this research.

In this paper, we define a cryptographic protocol as a mathematical object and security as a property of this object. Having formalized the concept of a secure protocol, we study its general properties. One of our main motivations for this work is the problem of combinations of protocols. It has been implicitly assumed in the literature that if two protocols are secure then these protocols can be performed sequentially without loss of security. This assumption turns out to be false more often than not. For example the seemingly harmless act of encrypting the same message using Rabin's encryption function under two different composite numbers is insecure. The message can be retrieved in polynomial time from the two encryptions [14]. The use of RSA with small exponent has similar problems [15]. In our model we are able to show a class of secure protocols which is closed under sequential execution. We call protocols in this class **strongly secure**.

Finally, we provide strongly secure solutions to some of the simpler problems discussed above. Solutions to more complex problems typically use these simpler protocols as subroutines. For example the coin flipping protocol is used in practically all solutions to the mental poker problem. Our results, while leaving open the problem of finding secure solutions to the harder class of transaction problems, increases our confidence in the use of simpler protocols as subroutines to more complex protocols.

2. The Protocol Environment or Model of Computation.

We think of a protocol as occurring between two Probabilistic Turing Machines (PTM's) A and B which operate synchronously. Each PTM has, besides its computation tape, a one-way infinite tape for incoming messages. We call this tape the "mailbox" of the machine. The PTM's communicate by writing into each others mailbox (Fig. 1).

We call such a system a CPTM (for Communicating Probabilistic Turing Machines). The mailbox tape symbols are digits , unary minus , letters, punctuation marks , and an end-of-message marker.

The PTMs have the capacity of reading a symbol from its mailbox at the same time as the symbol is being written. This convention is not essential to the model and any of the common resolutions of concurrent write-read will do.

CPTMs satisfy an "independence condition" which is stated as follows:

Independence Condition for CPTMs.

For all j , the conditional distribution of A's (B's) j^{th} . message given the prior messages between A and B is independent of the state of B (A) at the time of the message.

The Independence Condition holds for CPTMs because all communication in a CPTM occurs via mailboxes.

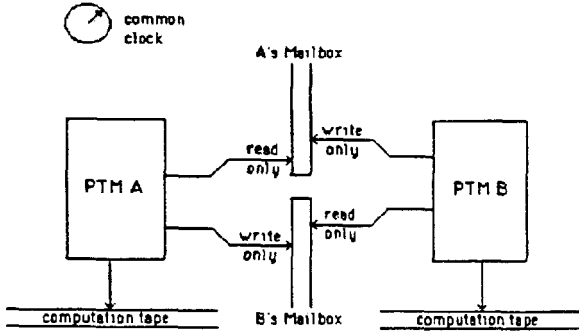


Figure 1. Communicating Probabilistic Turing Machines.

Our model will assume that factoring large integers is hard:

Definition - A Blum Integer is a composite $N = P \cdot Q$ where both P and Q are congruent to 3 mod 4 and the length of P is equal to the length of Q .

Assumption - Factoring Blum Integers is hard: For every poly-time probabilistic Turing Machine M , and any polynomial p , the probability that machine M factors a random n -bit Blum Integer is asymptotically less than $\frac{1}{p(n)}$.

3. Protocols Under the Assumption that Factoring Large Integers is Hard.

We now turn to the study of CPTM's whose parts A and B have computed and interchanged keys N_A and N_B with the following properties:

(let N be the key)

- i) The Jacobi symbol $(\frac{-1}{N}) = 1$.
- ii) N has exactly two distinct prime factors, both odd.
- iii) if z is a quadratic residue modulo N , then there exists roots x and y of z with opposite Jacobi symbols.

From now on when we refer to a number being a (public) key, we mean a number having properties i-iii. Under the assumption that factoring Blum integers is hard, A and B can generate (factored) keys whose factorization can not be computed by the opposite party. We will later show that there exist protocols such that the parties to the CPTM can convince each other that N_A and N_B are keys without helping each other factor the key. Note that not all numbers satisfying properties i-iii are Blum integers. We choose this

definition of public key because we know of no secure protocol by which the parties can convince each other that N_A and N_B are Blum integers. On the other hand these properties are enough to solve the problems typically solved by protocols using Blum integers.

4. Initialization of a CPTM.

The initial input to both A and B is a positive integer n , called the "security parameter" of the CPTM. The following steps are then carried on:

- Step 1 - A computes a key N_A of length n and writes it on B's mailbox.
- Step 2 - B reads N_A from its mailbox.
- Step 3 - B computes a key N_B of length n and writes it on A's mailbox.
- Step 4 - A reads N_B from its mailbox.

We start counting steps after initialization is completed. i.e. when we say the k -th step we mean the k -th click of the common clock after initialization.

We can not assume that A and B follow the initialization protocol. However, we will assume N_A, N_B are odd, composite, have length n , and satisfy property i) above. We can safely assume these properties because they are verifiable in polynomial time by a PTM without access to the factorization of N_A, N_B . We will later exhibit protocols through which A (B) can prove to B (A) that properties ii) and iii) hold without helping the opponent factor N_A (N_B).

5. The Definition of Cryptographic Protocol.

The concept of cryptographic protocol is used in various ways in the literature. The most common use of the term refers to two or more programs or computers with various communication capabilities. An alternative definition, proposed in [16], [17], [18] and [19] considers a protocol as a sequence of operators on messages. We will consider only 2-party protocols in which the parties are mutually distrusting. Our definition of cryptographic protocol refers not to the machines executing a communication but to the rules of such interaction. We will also ignore the problems of saboteurs or eavesdroppers on communication lines.

Definition - A protocol Π is a pair $[L, t(n)]$ ($L \in \mathcal{N}$, $t(n)$ a polynomially bounded function of n) and two sets of predicates

$$\begin{aligned} &P_i^A(m_1^A, \dots, m_i^A, m_1^B, \dots, m_{i-1}^B) \\ &P_i^B(m_1^A, \dots, m_i^A, m_1^B, \dots, m_i^B) \text{ for } i = 1, \dots, L. \end{aligned}$$

We denote the sequences $\{P_i^A\}$, $\{P_i^B\}$ by \vec{P}^A, \vec{P}^B respectively. L and $t(n)$ are called the "length" of Π and the "time between messages" of Π respectively. The semantics of this definition is as follows: m_i^A is the state of B's mailbox at time $(2i-1)t(n)$; m_i^B is the state of A's mailbox at time $2it(n)$. It is the responsibility of A to see that P_i^A is true for all i . It is the responsibility of B to see that P_i^B is true for all i .

The sequence $(m_1^A, m_1^B, \dots, m_L^A, m_L^B)$ is called a **conversation** between A and B. The reason we define m_i^A, m_i^B as states of mailboxes rather than as messages is that the former is always defined whereas the latter may not exist if one of the parties does not follow the protocol. As a consequence of this definition we may define the probability distribution φ_π of $(m_1^A, \dots, m_L^A, m_1^B, \dots, m_L^B)$ for fixed PTM's A and B.

We will typically leave $t(n)$ unspecified and argue simply that all computations necessary between messages can be done in probabilistic polynomial time. If A and B are fixed PTM's and Π is a protocol we denote the ordered triple (Π, A, B) by $\Pi(A, B)$.

Let the symbol \wedge stand for logical AND. We define the predicates $P^A(m_1^A, \dots, m_L^A, m_1^B, \dots, m_L^B)$, $P^B(m_1^A, \dots, m_L^A, m_1^B, \dots, m_L^B)$ as $\bigwedge_i P_i^A$ and $\bigwedge_i P_i^B$ respectively.

Protocol 1: Verifying that N_A satisfies property iii : if z is a quadratic residue modulo N_A , then there exists roots x and y of z with opposite Jacobi symbols.

Blum [1] proposed the following protocol :

For $i := 1$ to 100

1. A sends to B a random quadratic residue $x_i \pmod{N_A}$.
2. B sends to A $b_i = 1$ or -1 at random.
3. A sends to B a root of $x_i \pmod{N_A}$ with Jacobi symbol b_i .

If N_A satisfies property iii, then A will always be able to perform step 3. Otherwise, the probability that A can always respond at step 3 is $\leq 2^{-100}$.

In our formalism this protocol is written as follows :

Π_1 : Do 100 times the following protocol

$$P_1^A = (m_1^A \in Z_{N_A})$$

$$P_1^B = (m_1^B \in \{-1, 1\})$$

$$P_2^A = (\text{if } P_1^B \text{ then } (m_2^A)^2 = m_1^A \pmod{N_A} \text{ and } (\frac{m_2^A}{N_A}) = m_1^B)$$

$$P_2^B = (m_2^B = \text{"thanks"})$$

Notice that this protocol does not tell B how to behave in order to obtain proof that N_A has property iii. (Whereas Blum's version explicitly states how the parties should choose their messages). However, we can show that there exists a poly-time PTM B which follows the protocol such that, after the protocol, either A has been caught cheating, or the probability that N_A satisfies iii is $\geq 1 - 2^{-100}$. Throughout this paper we take the position that a protocol merely allows the parties to behave so as to achieve the desired goal, it cannot force them to do so.

6. Security.

We must first develop some notation.

Definition - A key-generator with input n generates a random factored n -bit Blum integer.

Definition - A poly-time PTM A is an honest player for protocol Π if:

- 1) its first step is a call to a key generator which returns (P_A, Q_A)
- 2) it goes through the initialization process with $N_A = P_A \cdot Q_A$
- 3) for all poly-time PTM B the probability of $P^A(m_1^A, \dots, m_L^A, m_1^B, \dots, m_L^B)$ is 1.

Since PTM's have limited computational power it is possible that there exists no honest player for a particular protocol. This motivates the following definition :

Definition - A protocol Π is **A-feasible** if there exists an honest player A for Π . We define **B-feasible** similarly. A protocol is **feasible** if it is A-feasible and B-feasible.

Definition - We say a protocol is **A-secure** if there exists an honest player A for Π such that for all poly-time PTM B the probability that B factors N_A goes to zero with n . The definition of **B-security** is analogous. We say a protocol is **secure** if it is both A-secure and B-secure.

We now define the notion of a **simulatable** player. This notion is essential for proving security of protocols. We want to put in precise terms the intuitive notion that if machine B can simulate the behavior of machine A in the protocol, then it is not possible for A to have released enough information for B to factor A's key.

There are a number of alternative definitions for the notion of simulator. Do we allow a machine which is simulating A to look at B's coin-tosses? Precisely what is to be simulated? In [4] (henceforth called the GMR model), simulators are considered which simply attempt to produce a sequence of messages with the same probability distribution as the actual conversation between A and B. That is, no attempt is made in that model to duplicate the environment in which a conversation between A and B takes place. For example, if B sends a random quadratic residue $x \bmod N_A$ to A, and A replies with a random square root of x modulo N_A , then it is easy to produce conversations with the same probability distribution as the actual conversation between A and B. To do this a machine M simply computes a random number x modulo N_A and lets $x^2 \bmod N_A$ be the message from B to A and x be the message from A to B. In the GMR model A is said to **release 0 knowledge** to B. On the other hand, we know that A has a chance of at least $\frac{1}{2}$ of releasing the factorization of N_A in this protocol (this is Rabin's Oblivious Transfer Protocol [8]). This awkward problem in the GMR model has no further consequence since, in that model as in this one, we are really interested in machines A which release no information to any machine B. For example, suppose B' is as B above except that it sends A the factorization of N_A if it obtains it. Then it is clear that there is no machine M which simulates A against B' unless factoring is in RP. Hence A does release knowledge to B', even though it does not release knowledge to B. In the GMR model machine A is said to **release 0 knowledge** if for all machines B, it releases 0 knowledge to B.

More serious drawbacks of the definitions in the GMR model are that i) it does not seem to go beyond the obvious statement that A releases no knowledge if and only if no machine B can put knowledge in the communication tape after a conversation (and hence it does not seem to provide a tool for the construction of 0-knowledge protocols) ; and ii) it is not clear whether or not the sequential execution of a polynomial

number of 0-knowledge protocols is still 0-knowledge (concatenation of protocols is a major goal in this model).

We will require that a simulator for machine A against machine B not only produce a possible conversation with the same distribution as conversations between A and B, but that it does so **while duplicating the interaction that B has with A**. It would be too restrictive to require a simulator to do this **all the time**, since it seems that in that case a simulatable machine A could not make use of the factorization of its key. Thus we relax this condition by requiring that a simulator succeeds in simulating A **with a constant probability greater than 0**. In addition to this we must require that a simulator realizes whether or not it succeeded in simulating A, otherwise simulatable protocols turn out to be unconcatenable. We now formalize these definitions.

Let Π be a protocol and B a player. Let A be an honest player which generates a random n-bit Blum integer N_A for a key. Recall that an honest player A will have put m_1^A in B's mailbox by time $(2i - 1) \cdot t(n)$. Let S be a procedure which, when called by B at time $(2i - 1) \cdot t(n)$ returns a message m_i^S . Let B[S] be machine B except that at step $(2i - 1) \cdot t(n)$ (after initialization) of B, B[S] calls S and sets $m_i^A = m_i^S$. We also give B[S] some extra power as follows: at any time B[S] may return to an earlier configuration and restart the computation from there. However, we will require that B[S] run in random polynomial time. We also require that S and B satisfy the Independence Condition for CPTMs defined in section 2. Notice that B[S] is a poly-time PTM with input n, N_A . Thus, if N_A is a random n-bit Blum integer, the probability that B[S] can factor N_A is asymptotically 0 by assumption.

We define σ_n to be the probability distribution of $(m_1^S, \dots, m_L^S, m_1^B, \dots, m_L^B)$ in B[S] with security parameter n . Recall that φ_n is the distribution of conversations between A and B with security parameter n .

Definition - Let Π be a protocol with A an honest player. We say S is an **A-simulator** for Π if for n sufficiently large, for all players B, and for all pairs (N_A, N_B) , machine B[S] satisfies the following conditions:

- i) the probability ρ of $P^A(Y)$ given N_A, N_B is a constant greater than 0 and the event $P^A(Y)$ is independent of B's coin tosses.
- ii) S decides $P^A(x)$ with error probability 0 for all x .
- iii) $\sigma_n(x \mid P^A(Y), N_A, N_B) = \varphi_n(x \mid N_A, N_B)$ for all x .

where Y is a random variable which assumes values $(m_1^S, \dots, m_L^S, m_1^B, \dots, m_L^B)$ in B[S].

Since B[S] can return to earlier configurations we see that the constant ρ can be made exponentially close to 1. For example if $\rho = \frac{1}{2}$ for machine B[S] we can define another machine B'[S] which runs B[S] and if $P^A(Y)$ is not true, runs B[S] again. The probability that $P^A(Y)$ is true for B'[S] is now $\frac{3}{4}$. In general, if we allow for k trials of B[S], the probability of $P^A(Y)$ is $1 - (\frac{1}{2})^k$.

Definition - Let Π be a protocol and A an honest player for Π . We say A is **simulatable** if there exists an A-simulator for A.

Theorem 1. Let Π be a protocol and A an honest player for Π . If A is simulatable and S is a simulator for A then for any pair of keys (N_A, N_B) , the probability that B factors N_A given (N_A, N_B) is less than a constant times the probability that B[S] factors N_A . The constant is independent of the keys.

Proof: Fix A, B, N_A , N_B . Let S be an A-simulator for $\Pi(A, B)$. Let X, Y be random variables which assume values $(m_1^A, \dots, m_L^A, m_1^B, \dots, m_L^B)$ in $\Pi(A, B)$, and $(m_1^S, \dots, m_L^S, m_1^B, \dots, m_L^B)$ in B[S] respectively. Let E be the event that B factors N_A in Π . Let E' be the event that B[S] factors N_A . For the remainder of the proof all probabilities are conditional on the values of N_A, N_B .

Let Ω be the message space and $x = (m_1^A, \dots, m_L^A, m_1^B, \dots, m_L^B) \in \Omega^{2L}$. Recall $\varphi_n(x) = \sigma_n(x|P^A(Y))$ and note that the independence condition on S implies that for all x, $\text{Prob.}(E|X = x) = \text{Prob.}(E' | Y = x)$.

Thus,

$$\begin{aligned}
 & \text{Prob.}(E) \\
 &= \int_x \varphi_n(x) \cdot \text{Prob.}(E | X = x) \\
 &= \int_x \sigma_n(x|P^A(Y)) \cdot \text{Prob.}(E' | Y = x) \\
 &= \int_x \sigma_n(x|P^A(Y)) \cdot \text{Prob.}(E' | (Y = x) \wedge P^A(Y)) \\
 &= \text{Prob.}(E' | P^A(Y)) \\
 &\leq \frac{\text{Prob.}(E')}{\text{Prob.}(P^A(Y))} \\
 &\leq \frac{\text{Prob.}(E')}{\rho}.
 \end{aligned}$$

The third equality is justified by the fact that if $P^A(Y)$ is true then $Y = x$ implies $P^A(Y)$.

Corollary 1. Let Π be a protocol and A an honest player for Π . If A is simulatable then Π is A-secure.

Simulatability is a strong requirement. It is conceivable that a protocol is secure without being simulatable. This motivates the following definition :

Definition - A protocol Π is **strongly secure** if there exist honest simulatable players A and B for Π .

Strongly secure protocols have the desirable property that they release no partial information about the factorization of the player's private keys. This results in strongly secure protocols being "concatenable". That is, a polynomial number of strongly secure protocols can be run under the same keys. We now formalize this idea.

Definition - Let A_1, A_2 be honest players for protocols Π_1, Π_2 respectively. The machine $A_3 = A_1|A_2$ is defined by the following rules :

A_3 runs as A_1 until A_1 halts.
 Then A_3 runs as A_2 except that, rather than obtaining N_A from the key generator, it skips the initialization routine using the keys N_A, N_B known to A_1 instead.

Definition - Let $\Pi_1 = (L_1, t_1(n), \bar{P}^{A_1}, \bar{P}^{B_1})$ and $\Pi_2 = (L_2, t_2(n), \bar{P}^{A_2}, \bar{P}^{B_2})$ be two protocols. We define the **concatenation** $\Pi_1 \% \Pi_2$ of Π_1, Π_2 as follows:

$$\Pi_1 \% \Pi_2 = (L_3, t_3(n), \bar{P}^{A_3}, \bar{P}^{B_3})$$

where

$$L_3 = L_1 + L_2$$

$$t_3(n) = \max\{t_1(n), t_2(n)\}$$

$$P_i^{A_3} = P_i^{A_1} \text{ for } i = 1, \dots, L_1$$

$$P_{L_1+i}^{A_3} = P_i^{A_2} \text{ for } i = 1, \dots, L_2$$

$$P_i^{B_3} = P_i^{B_1} \text{ for } i = 1, \dots, L_1$$

$$P_{L_1+i}^{B_3} = P_i^{B_2} \text{ for } i = 1, \dots, L_2$$

In other words, concatenation of two protocols is simply the concatenation of the two sequences of predicates.

Now we are ready to show that simulatable protocols are concatenable. Even though the statement is intuitively true, the proof is somewhat technical.

Figure 2.a) depicts a CPTM composed of a PTM B and an adversary $A_1|A_2$ where A_1, A_2 are honest simulatable players for two protocols Π_1, Π_2 respectively. A machine R_B (the "restriction of B to Π_1 ") is defined from machine B in the following way: R_B behaves as B until A_2 starts executing, at which point it halts. Figure 2.b) depicts the same PTM B but with A_1, A_2 replaced by simulators S_1, S_2 respectively. Figure 2.c) depicts PTM B with adversaries S_1 and then A_2 . The random variables $X_1, X_2, V_1, V_2, Z_1, Z_2$ represent the messages between A_1-B , A_2-B , S_1-B , S_2-B , S_1-B , A_2-B respectively in the given configurations.

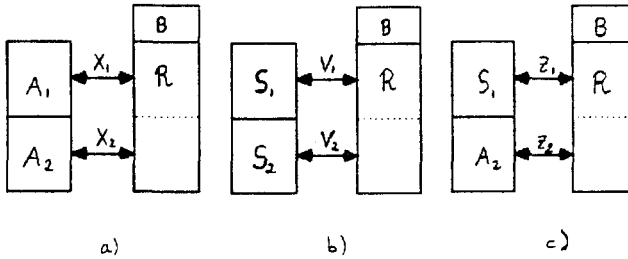


Figure 2.

Theorem 2. Let Π_1, Π_2 be A-secure protocols with honest simulatable players A_1, A_2 . Then $A_3 = A_1|A_2$ is an honest simulatable A-player for $\Pi_3 = \Pi_1 \% \Pi_2$.

Proof: Honesty of $A_1|A_2$ follows immediately from the construction of $A_1|A_2$. Thus we need only show simulatability. We must show that for n large enough, for all PTM B, for all a, b, N_A, N_B ,

$$Prob.(X_1 = a, X_2 = b) = Prob.(V_1 = a, V_2 = b \mid P^{A_1}(V_1) \wedge P^{A_2}(V_2)) \quad (*)$$

Let n , the size of the public keys, be large enough so that the simulatability conditions hold for both A_1 and A_2 with simulators S_1 and S_2 respectively. Fix a, b, N_A, N_B . From now on all probabilities are taken conditioning on the values of N_A, N_B . If $\neg P^{A_1}(a)$ or $\neg P^{A_2}(b)$ then both sides of (*) are zero. Suppose $P^{A_1}(a)$ and $P^{A_2}(b)$. If $P^{A_1}(a)$ and $Prob.(V_1 = a) = 0$ then, by the simulatability conditions, both sides of (*) are 0.

Suppose $\text{Prob.}(V_1 = a) > 0$.

We first show that $\text{Prob.}(X_1 = a, X_2 = b) = \text{Prob.}(Z_1 = a, Z_2 = b \mid P^{A_1}(Z_1))$. Note that this is implied by the 2 equations

$$(i) \text{Prob.}(X_2 = b \mid X_1 = a) = \text{Prob.}(Z_2 = b \mid Z_1 = a)$$

and

$$(ii) \text{Prob.}(X_1 = a) = \text{Prob.}(Z_1 = a \mid P^{A_1}(Z_1)).$$

The first equation holds by the Independence Condition for CPTMs. Equation (ii) holds by simulatability.

Now we show that

$$\text{Prob.}(Z_1 = a, Z_2 = b \mid P^{A_1}(Z_1)) = \text{Prob.}(V_1 = a, V_2 = b \mid P^{A_1}(V_1) \wedge P^{A_2}(V_2)) \quad (**).$$

This equation is implied by the two equations

$$(iii) \text{Prob.}(Z_1 = a \mid P^{A_1}(Z_1)) = \text{Prob.}(V_1 = a \mid P^{A_1}(V_1) \wedge P^{A_2}(V_2))$$

$$(iv) \text{Prob.}(Z_2 = b \mid Z_1 = a) = \text{Prob.}(V_2 = b \mid P^{A_2}(V_2) \wedge V_1 = a).$$

Equation (iii) can be shown as follows:

$$\begin{aligned} \text{Prob.}(V_1 = a \mid P^{A_1}(V_1) \wedge P^{A_2}(V_2)) &= \frac{\text{Prob.}(V_1 = a \wedge P^{A_1}(V_1) \mid P^{A_2}(V_2))}{\text{Prob.}(P^{A_1}(V_1) \mid P^{A_2}(V_2))} \\ &= \frac{\text{Prob.}(V_1 = a \wedge P^{A_1}(V_1))}{\text{Prob.}(P^{A_1}(V_1))} \\ &= \frac{\text{Prob.}(V_1 = a)}{\text{Prob.}(P^{A_1}(V_1))} \\ &= \text{Prob.}(V_1 = a \mid P^{A_1}(V_1)). \end{aligned}$$

The second equality holds because the event $P^{A_2}(V_2)$ is independent of the events $(V_1 = a \wedge P^{A_1}(V_1))$ and $P^{A_1}(V_1)$ by definition of simulatability. The third and fourth equalities hold because we have assumed $P^{A_1}(a)$.

Equation (iv) holds because S_2 is a simulator for A_2 and for all machines B . In particular, S_2 is a simulator for A_2 executing the protocol against a machine M which is $B[S_1]$ with the condition that M chooses its coin tosses randomly and uniformly only among those which yield $Z_1 = a$. Machine M is depicted in Figures 3.a) and 3.b) playing against A_2 and S_2 respectively. Note that

$$\text{Prob.}(V_2 = b \mid P^{A_2}(V_2) \wedge V_1 = a) = \text{Prob.}(U_2 = b \mid P^{A_2}(U_2)) = \text{Prob.}(T_2 = b) = \text{Prob.}(Z_2 = b \mid Z_1 = a).$$

Combining equations (*) and (**) completes the proof of the theorem QED .

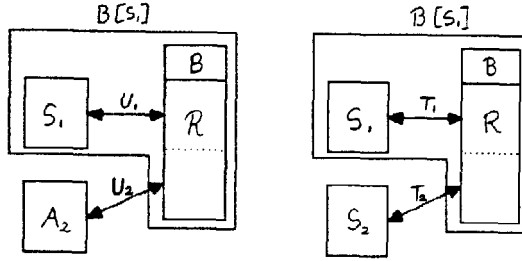


Figure 3.

Theorem 3. Protocol 1 is a strongly secure protocol.

Proof: Protocol 1 is a concatenation of 100 protocols

$$\Pi_1: P_1^A = (m_1^A \in Z_{N_A})$$

$$P_1^B = (m_1^B \in \{-1, 1\})$$

$$P_2^A = (\text{if } P_1^B \text{ then } (m_2^A)^2 = m_1^A \pmod{N_A} \text{ and } (\frac{m_2^A}{N_A}) = m_1^B)$$

$$P_2^B = (m_2^B = \text{"thanks"})$$

Notice that Bob does not use his key, therefore we need only be concerned about the protocol being A-secure. By theorem 2, it is enough to display a simulatable algorithm for A in Π . Let A's algorithm be the following:

message 1:

Send a random quadratic residue $m_1^A \in Z_{N_A}$.

message 2:

If m_1^B is 1 or -1 then send the root of m_1^A with Jacobi symbol m_1^B else send "you are cheating".

Then the following algorithm is an A-simulator:

message 1:

Choose a random number $x \in Z_{N_A}$ and send $m_1^S = x^2 \pmod{N_A}$.

message 2:

if m_1^B is 1 or -1 and the Jacobi symbol of x modulo N_A is m_1^B then send $m_2^S = x \pmod{N_A}$. If m_1^B is not 1 or -1 then flip a fair coin. If the outcome is heads send $m_2^S = \text{"you are cheating"}$. If the outcome is tails, machine B[S] returns to the initial configuration and the simulation is repeated.

The last instruction of machine B[S] may seem intriguing at first glance. However, it is necessary in order to satisfy properties i) and iii) of a simulator. The problem is the following: if B does not send $m_1^B = \pm 1$ then S has a chance of only $\frac{1}{2}$ of satisfying P^A . Therefore, if S simply responds "you are

cheating" when B sends $m_1^B \neq \pm 1$, the conditional probability of a conversation x given that S satisfies P^A is biased towards those conversations in which B sends $m_1^B \neq \pm 1$.

If we consider B's messages as questions to A (or S) then the problem is that all questions are easily answered by A, whereas the probability that S can answer questions may not be the same for all questions. Machine B[S] must incorporate instructions to homogenize the hardness of replying to B.

Having said this, verification that B[S] satisfies properties i) to iii) of a simulator is easy and is left to the reader.

7. Strongly Secure Solutions to Some Cryptographic Problems.

In this section we provide strongly secure solutions to some well-known cryptographic problems.

Protocol 2 - Coin Flipping Into a Well.

The purpose of this protocol is for Alice to give Bob a random bit. However, Alice must not know which bit she gave him until Bob displays the bit. Bob, on his part, cannot lie about which bit he got. Since we have shown that Protocol 1 is strongly secure we may assume that N_A satisfies property iii) of a public key. The protocol is as follows:

$$\begin{aligned}\Pi_2 : P_1^A(m_1^A) &= (\text{"let's flip a coin into your well"}) \\ P_1^B(m_1^B) &= (m_1^B \in Z_{N_A}) \\ P_2^A(m_2^A) &= (m_2^A \in \{1, -1\}) \\ P_2^B(m_2^B) &= ((m_2^B)^2 = m_1^B \bmod N_A)\end{aligned}$$

Alice's and Bob's algorithms are as follows:

```
Alice :
message 1:
    send  $m_1^A = \text{"let's flip a coin into your well"}$ 

message 2:
    send  $m_2^A = +1 \text{ or } -1 \text{ at random.}$ 

Bob :
message 1 :
    choose  $x$  at random and send  $m_1^B = x^2 \bmod N_A$ .

message 2 :
    send  $m_2^B = x$ .
```

The value of the coin-flip is $(\frac{x}{N_A}) \cdot b$. Bob may display the value of the coin-flip by displaying the root of x^2 that he knows. Until he displays the coin-flip at message 2 (m_2^B), Alice has no idea of what the value of the coin-flip is. If Alice is honest she can be sure that Bob cannot lie about the bit he got because he knows at most one root x of $x^2 \bmod N_A$ (and, of course, $-x$).

This protocol is simulatable because neither party uses the factorization of their keys. Assuming that Bob knows a root x , the probability that the coin-flip is 1 is the same as the probability that the coin-flip is -1. If Bob is honest, he can be sure that the coin-flip is unbiased because, from Alice's point of view, the

probability that x has Jacobi symbol 1 is the same as the probability that x has Jacobi symbol -1.

An important observation, which will be used in protocol Π_4 , is that a simulator S can apriori choose the value of the coin-flip provided it chooses it at random. To do this S sends $m_2^S = \pm 1$ at random. If the outcome of the coin-flip is different from the one S wants, machine $B[S]$ can simply restart computation at the beginning of the protocol execution. In random polynomial time, S can obtain the flip it originally chose.

Protocol 3 - Generating a random element in Z_{N_A} .

The coin-flipping Protocol 2 can be used $n = |N_A|$ times to generate a random element in Z_{N_A} .

Protocol 4 - Verifying that N_A has exactly 2 prime factors, both odd.

This problem has been studied extensively by mathematicians. To this date, an efficient algorithm to determine the number of distinct prime factors of a composite number (the index of the number) has not been found. It is possible that no such algorithm exists. It is a remarkable achievement of the research on interactive proof systems that a proof that the index of a composite number can be shown to be 2 by an omniscient party without releasing any additional information about the composite number.

The crucial observation for this solution is due to Adleman [20]. He suggested using the fact that if N_A has more than two prime factors, then at most $\frac{1}{8}$ of the numbers in Z_{N_A}' are quadratic residues. The protocol uses Protocol 3 to generate M random numbers with Jacobi symbol 1 in Z_{N_A} . Having done this, Alice reveals a square root of each of the numbers which is a quadratic residue. Bob accepts the number N_A if Alice reveals at least αM square roots. The parameters M and α are chosen so as to obtain a negligible probability of error at the minimum possible cost M . This solution has a two-sided error probability. It is possible for Alice to convince Bob that N_A has at most 2 prime factors when it in fact has more than 2, and it is possible for Alice to be unable to complete the proof even though N_A has exactly two prime factors. We now derive an approximation to the optimum value of α .

Let Y be the number of quadratic residues among M random numbers modulo N_A . Let p be the probability that a random number in Z_{N_A} is a quadratic residue. By the Central Limit Theorem (see any probability textbook, for example [21]) the random variable

$$Z = \frac{Y - pM}{\sqrt{Mp(1-p)}}$$

is asymptotically $\Phi(0,1)$ (normal with mean zero and standard deviation 1).

For M in the hundreds and $p \in [\frac{1}{8}, \frac{1}{4}]$, $\Phi(0,1)$ is a good approximation to the distribution of Z . From now on we compute probabilities under the assumption that Z has distribution $\Phi(0,1)$.

Let ε_1 be the probability that Bob rejects N_A when N_A has exactly two prime factors. Let ε_2 be the probability that Bob accepts N_A when N_A has exactly three prime factors. Then, if N_A has exactly two prime factors we have

$$\begin{aligned}
\varepsilon_1 &= P(Y < \alpha M) \\
&= P(Z < \frac{(\alpha - .25)M}{\sqrt{M(.25)(.75)}}) \\
&= P(Z < \frac{(4\alpha - 1)\sqrt{M}}{\sqrt{3}}) \\
&= \int_{-\infty}^{\frac{(4\alpha - 1)\sqrt{M}}{\sqrt{3}}} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}} dt \quad (I)
\end{aligned}$$

Similarly, if N_A has exactly three prime factors we have

$$\begin{aligned}
\varepsilon_2 &= 1 - P(Z < \frac{(8\alpha - 1)\sqrt{M}}{\sqrt{7}}) \\
&= \int_{-\infty}^{\frac{-(8\alpha - 1)\sqrt{M}}{\sqrt{7}}} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}} dt \quad (II)
\end{aligned}$$

Lemma 1.

$$\text{If } x \text{ is negative then } \int_{-\infty}^x \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}} dt < \frac{-2e^{-\frac{x^2}{2}}}{x\sqrt{2\pi}}.$$

Proof: $x < 0$ and $t \leq x$ implies $-t^2 \leq -xt$.

$$\text{Thus } \int_{-\infty}^x \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}} dt < \int_{-\infty}^x \frac{e^{-\frac{xt}{2}}}{\sqrt{2\pi}} dt = \frac{-2e^{-\frac{x^2}{2}}}{x\sqrt{2\pi}}.$$

Theorem 4. For all values of α , $e^{\frac{-M}{74}} < \text{Max}\{\varepsilon_1, \varepsilon_2\} < e^{\frac{-M}{75}}$ asymptotically. We can achieve an error probability in this range if we let $\alpha = \frac{\sqrt{21} - 1}{20}$.

Proof: Since we seek to minimize $\text{Max}\{\varepsilon_1, \varepsilon_2\}$, it is clear that the optimal value of α is somewhere between $\frac{1}{8}$ and $\frac{1}{4}$. Thus $(4\alpha - 1) < 0$. Using the lemma, and substituting in $(4\alpha - 1)\sqrt{M}/\sqrt{3}$ and $-(8\alpha - 1)\sqrt{M}/\sqrt{7}$ for x in (I) and (II) respectively, we get

$$\varepsilon_1 < \frac{-2\sqrt{3} e^{-\frac{(4\alpha - 1)^2 M}{6}}}{(4\alpha - 1)\sqrt{2\pi M}}$$

and

$$\varepsilon_2 < \frac{2\sqrt{7} e^{-\frac{(8\alpha - 1)^2 M}{14}}}{(8\alpha - 1)\sqrt{2\pi M}}.$$

Thus both ε_1 and ε_2 are bounded above by functions of the form $a \frac{e^{-bM}}{\sqrt{M}}$. Since the parameter b dominates the expression for the bound, we would like b to be the same for ε_1 and ε_2 , i.e. we want $\frac{(4\alpha - 1)^2}{6} = \frac{(8\alpha - 1)^2}{14}$. Solving for α we get $\alpha = \frac{\sqrt{21} - 1}{20}$, which makes $b \approx .01339 > \frac{1}{75}$. Thus $\text{Max}\{\varepsilon_1, \varepsilon_2\} < e^{\frac{-M}{75}}$.

A similar argument, involving the (asymptotic) inequality $e^{-\frac{t^2}{2}} > -t e^{-\frac{t^2}{(2-\beta)}}$ for $2 > \beta > 0$ and $t < 0$ shows that $\text{MAX}\{\epsilon_1, \epsilon_2\}$ is asymptotically greater than $e^{-\frac{M}{74}}$. Q.E.D.

We have shown that, even though, this protocol achieves exponentially small probability of error, we must use M in the thousands in order to achieve truly negligible probability of error.

This protocol requires the communication of a very large number of bits. It is expensive in communication and computation. This is also the fastest known protocol for this problem. Goldwasser, Micali, and Rackoff [4] have an elegant but expensive 0-knowledge interactive proof by which Bob can prove to Alice that he knows a root of a quadratic residue modulo her key. Using this technique a 0-knowledge protocol for this problem can be constructed which is essentially a hundred times as expensive as our protocol. Protocols for harder problems, e.g. Blum's certified mail protocol, may require the execution of this protocol hundreds or thousands of times for different keys. This illustrates the practical need for protocols which use a single key.

It would be straight-forward but cumbersome to write this protocol in our formalism. Instead, we write it out in a hybrid notation and argue informally that it is simulatable.

Π_4 : Do 3000 times

- i) Execute Protocol Π_3 to generate a random number x in Z_{N_A} .
- ii) If the Jacobi symbol of $x \bmod N_A$ is 1 then Alice sends the message "non-residue" or a square root of $x \bmod N_A$.

Theorem 5. Π_4 is strongly secure.

Proof: The reason that this needs to be proven is that it does not follow immediately from Theorem 2. This is because Π_4 is not a concatenation of strongly secure protocols. However, if Alice follows the algorithm given for Π_2 and honestly executes instruction ii) of Π_4 then we can argue that Alice is simulatable.

We argue informally as follows: A simulator for Π_3 , the protocol which generates a random element in Z_{N_A} , can choose apriori what number is to be generated (see the note on this matter in the description of Π_2) provided it chooses it at random. Thus S can simulate A as follows: i) S flips a fair coin to decide whether the number generated in the simulation of Π_3 will have Jacobi symbol 1 or -1. If the number is to have Jacobi symbol -1 then S simply generates a random element with Jacobi symbol -1. If the number is to have Jacobi symbol 1 then S flips a fair coin to decide whether it will choose a quadratic residue or a quadratic non-residue. Then S generates a random element in $r \in Z_{N_A}$. If x in step i) of Π_4 is to be a non-residue then S sets $x = -r^2 \bmod N_A$. If x is to be a residue then S sets $x = r^2 \bmod N_A$. The reader can verify that x , chosen in this way, is indeed a random element in Z_{N_A} . If x is a quadratic residue then S knows a square root of x and thus can execute step ii) of Π_4 . Q.E.D.

Note that the properties of public key N_A are crucial in this proof. This is because if N_A is a public key then -1 modulo N_A is a non-residue with Jacobi symbol 1. If N_A was an arbitrary composite then this protocol would not be simulatable since there is no known effective algorithm to compute a non-residue with

Jacobi symbol 1 modulo an arbitrary composite. This completes the proof that Alice and Bob can convince each other that $N_A N_B$ are valid public keys without helping the opponent factor the key.

Protocol 5 - The Oblivious Transfer.

A strongly secure variant of Rabin's Oblivious Transfer, called "The Probabilistic Channel", has been implemented in [22] based on an earlier work on the Oblivious Transfer [6].

Acknowledgements.

The importance of studying cryptographic protocols in a rigorous way was made clear to us by Manuel Blum. He also guided us throughout this research with insight and valuable references. He, of course, bears no responsibility for the possible weaknesses and shortcomings of this model. Other good friends who worked with us on this problem include Tom Tedrick and Umesh Vazirani.

References.

1. M. Blum, Coin Flipping by Telephone, *Proc. IEEE COMPCON*, 1982, 133-137.
2. M. Luby, S. Micali and C. Rackoff, How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin., *24th. IEEE Annual Symp. on Foundations of Computer Science*, 1983, 11.
3. T. Tedrick, How to Exchange Half a Bit, *Proceedings of Crypto 83*, N.Y., 1984, 147.
4. S. Goldwasser, S. Micali and C. Rackoff, The Knowledge Complexity of Interactive Proof Systems, *17th. Annual ACM Symp. on Theory of Computing*, 1985.
5. M. Fischer, S. Micali and C. Rackoff, A Secure Protocol for the Oblivious Transfer, *Proceedings of Eurocrypt 84.*, 1984.
6. R. Berger, R. Peralta and T. Tedrick, *A Provably Secure Oblivious Transfer*, Dept. EECS, Univ. of California, Berkeley, Calif., 1983.
7. M. Blum, How to Exchange Secret Keys, *ACM Transactions on Computer Systems* 1, 2 (May 1983), 175-193.
8. M. Blum, *Three Applications of the Oblivious Transfer* : 1. *Coin Flipping by Telephone*, 2. *How to Exchange Secrets*, 3. *How to Send Certified Electronic Mail*, Dept. EECS, Univ. of California, Berkeley, Calif., 1981.
9. S. Even, O. Goldreich and A. Lempel, A Randomized Protocol for Signing Contracts, *Technical Report #233*, February 1982..
10. S. Fortune and M. Merritt, Poker Protocols, *Crypto 84*, 1984.
11. M. Yung, Cryptoprotocols : Subscription to a Public Key, the Secret Blocking and the Multi-Player Mental Poker Game., *Crypto 84*, 1984.
12. L. Blum, M. Blum and M. Shub, A Simple Secure Pseudo-Random Number Generator, *CRYPTO 82*, 1982.
13. S. Goldwasser and M. Blum, An Efficient Probabilistic Public-Key Encryption Scheme Which Hides All Partial Information., *Crypto 84*, 1984.

14. M. Blum, *A Potential Danger with Low-Exponent Modular Encryption Schemes: Avoid Encrypting Exactly the Same Message to Several People.*, U.C. Berkeley Computer Science Department, 1984.
15. J. Hastad, *On Using RSA with Low Exponent in a Public Key Network.*, MIT Computer Science Department, 1984.
16. D. Dolev, S. Even and R. Karp, *On The Security Of Ping-Pong Protocols*, *Proceedings of Crypto 82*, 1982.
17. D. Dolev and A. Yao, *On The Security Of Public Key Protocols*, *IEEE Transactions on Information Theory*, IT-30 (March 1983), 198.
18. M. Merritt, *Cryptographic Protocols* , Ph.D Thesis. Georgia Institute of Technology, GIT-ICS-83/06. 1983.
19. M. Merritt and P. Wolper, *States of Knowledge in Cryptographic Protocols.*, *Unpublished Manuscript.* , .
20. L. Adleman, *Private Communication through M. Blum.*, 1983.
21. K. Chung, *A Course in Probability Theory*, Academic Press, London, 1974.
22. R. Peralta and T. Tedrick, *The Probabilistic Channel*, *In preparation*, 1985.

CHEATING AT MENTAL POKER

Don Coppersmith
IBM Research
Yorktown Heights, NY 10598

We review the "mental poker" scheme described by Shamir, Rivest and Adleman [SRA]. We present two possible means of cheating, depending on careless implementation of the SRA scheme. One will work if the prime p is such that $p - 1$ has a small prime divisor. In the other scheme, the names of the cards "TWO OF CLUBS" have been extended by random-looking bits, chosen by the cheater.

Background

In 1979 Shamir, Rivest and Adleman [SRA] proposed a scheme for playing "mental poker," i.e. play a fair poker game over the telephone between two mutually suspicious players. As a corollary, their paper gave a practical method for exchanging secret information over a public channel. (This method of exchanging information is still viable, and nothing in this paper affects its usefulness.)

In their scheme, players A and B agree on a large prime p . They create a deck of cards c_i , $i = 1, 2, \dots, 52$, where, for example, c_1 might be the EBCDIC coding of the characters "TWO OF CLUBS". Player A creates two secret numbers a, \bar{a} , such that $a\bar{a} \equiv 1 \pmod{p}$; Player B similarly creates secret numbers b, \bar{b} . Player A shuffles the deck, encodes each card by raising to the a power \pmod{p} , and sends the deck to Player B . (At this point, B sees $c_{\pi(i)}^a \pmod{p}$, where π denotes the permutation or shuffle applied by A .) Player B selects five cards for A , say $c_{A1}^a \pmod{p}, \dots, c_{A5}^a \pmod{p}$, and returns them to A , who decodes them by raising to the \bar{a} power \pmod{p} . B also selects five cards for himself, and adds his own encryption by raising to the b power \pmod{p} . He sends the resulting cards, $c_{B1}^{ab} \pmod{p}, \dots, c_{B5}^{ab} \pmod{p}$, to A . In turn, A raises B 's cards to the \bar{a} power, obtaining $c_{Bi}^{a\bar{a}b} \equiv c_{Bi}^b \pmod{p}$, and returns them to B . Finally B raises these cards to the \bar{b} power \pmod{p} to obtain $c_{Bi}^{b\bar{b}} \equiv c_{Bi} \pmod{p}$, his own hand in the clear.

Thus is the hand dealt. Betting proceeds as usual. At the end of the game, the secret keys are revealed, so that the hands are made known to both sides.

Method 1: when $p-1$ has a small factor.

The first method of cheating is a generalization of the "quadratic residue" trick, due to Lipton [DDHHL].

Suppose that $p - 1$ is divisible by a small integer q , say $30 < q < 10^{12}$.

The multiplicative group of integers \pmod{p} is denoted by \mathbb{Z}_p^* . It is isomorphic to the additive group of integers $\pmod{p-1}$, \mathbb{Z}_{p-1} . (There are several isomorphisms available, and we can select one by selecting a generator g of the multiplicative group.) For each integer q dividing $p - 1$ there is a projection from \mathbb{Z}_{p-1} onto \mathbb{Z}_q . Composing these two maps, to each $x \neq 0 \pmod{p}$ we can associate an element \pmod{q} , which we will call $\log x \pmod{q}$, suppressing

the dependence on g . The Pohlig-Hellman technique ([PH], attributed by them to Roland Silver) enables us to compute $\log x \pmod{q}$ for the price of $O(\log p + \sqrt{q})$ multiplications \pmod{p} . For q in the range given, this is a feasible amount of computation.

Suppose Player B sees the cards before they are encrypted. Then he can determine $\{\log c_i \pmod{q}, 1 \leq i \leq 52\}$. Now he receives the shuffled and encrypted deck from A . Again, he determines $\{\log(c_{\pi(i)}) \pmod{q} \approx a \log c_{\pi(i)} \pmod{q}\}$. By comparing the distributions of the logarithms, before and after encryption, B can usually determine the value of $a \pmod{q}$. Thus he can recover $\{\log c_{\pi(i)} \pmod{q}\}$. This gives him some information about the permutation π : he can tell which cards are which, up to ambiguities caused when two logarithms are the same: $\log c_i \pmod{q} = \log c_j \pmod{q}$. The expected number of uniquely determined cards is about $52(e^{-51/q})$; for $q > 30$ one expects to have at least nine cards uniquely determined.

Finally, if we choose our prime p uniformly at random, we will have some prime q , $30 < q < 10^{12}$, dividing $p - 1$ about eighty-seven percent of the time.

Conclusion 1: If you don't want cheating, choose your primes p to be of the form $p = 2q + 1$, q prime, so that the cheater can only tell the difference between quadratic residues and non-residues. Also, append bits so that all the cards are quadratic residues, to block even that information from the cheater.

Method 2: when the cards are padded by random bits.

The string "THREE OF DIAMONDS" in EBCDIC is very short: only seventeen characters or 136 bits. Our prime p cannot be this short, because efficient techniques exist for finding logarithms modulo primes this small [WM], [Adl], [COS]. So suppose the EBCDIC strings are padded out with random bits, in accordance with good cryptographic practice. (Note: the original paper [SRA] did not suggest such padding.) Suppose these bits occupy half the description of the cards.* Suppose also that Player B is allowed to select these "random bits". Then he can cheat.

Let the i^{th} card be given by $c_i = s_i + r_i < p$, where s_i is the EBCDIC coding of the card's name in English, left-adjusted in the representation of the integer, and r_i is the "random" portion, constricted by $0 \leq r_i \leq \sqrt{p}$.

Player B fixes the representation of c_1 as "TWO OF CLUBS" padded with truly random bits. Now for each i , $2 \leq i \leq 26$, B tries to select r_{2i-1}, r_{2i} so that the resulting integers c_{2i-1}, c_{2i} satisfy

$$\begin{aligned} c_1^i c_{2i-1}^{-1} &\equiv c_{2i} \pmod{p} \\ (c_1^i s_{2i-1} \pmod{p}) + (c_1^i \pmod{p}) r_{2i-1} &= s_{2i} + r_{2i} + tp, \end{aligned}$$

where r_{2i-1}, r_{2i} and t are unknown integers less than \sqrt{p} . This is just a linear diophantine equation, easily solved, for example, by a basis reduction algorithm; see [Lag], [LLL] for the techniques involved.

* An interesting problem remains: what if the "random bits" occupy only 1/3 or 1/4 of the description? Can a similar scheme be implemented?

Now A shuffles and encrypts the deck, and sends the entire deck to B . Recall that B sees $c_{\pi(j)}^a \pmod{p}$. Notice that since $c_1^i c_{2i-1}^i \equiv c_{2i}^i \pmod{p}$, then the same relation holds among the encrypted cards: $(c_1^a)^i (c_{2i-1}^a)^i \equiv (c_{2i}^a)^i \pmod{p}$. So B tries each of $52 \times 51 = 2652$ ordered pairs of cards in the encrypted shuffled deck, computing $(c_{\pi(j)}^a)^2 (c_{\pi(k)}^a)^i \pmod{p}$ and comparing the results to the remaining 50 cards. On finding a match, $(c_{\pi(j)}^a)^2 (c_{\pi(k)}^a)^i \equiv (c_{\pi(l)}^a)^i \pmod{p}$, Player B has probably identified three cards: $\pi(j) = 1$, $\pi(k) = 3$, $\pi(l) = 4$. Now for $3 \leq i \leq 26$, $1 \leq m \leq 52$, $m \neq j, k, l$, compute $(c_{\pi(j)}^a)^i (c_{\pi(m)}^a)^i \pmod{p}$ and compare to the remaining cards; each match $(c_{\pi(j)}^a)^i (c_{\pi(m)}^a)^i \equiv (c_{\pi(n)}^a)^i \pmod{p}$, gives two more cards $\pi(m) = 2i - 1$, $\pi(n) = 2i$. At the cost of a few thousand multiplications \pmod{p} , B has recovered the permutation π , and can now select both hands quite maliciously.

Conclusion 2: If you're going to have "random padding," make sure your opponent doesn't select the random numbers.

Conclusion 3: The protocol is fairly fragile in the sense that seemingly innocuous changes (selection of p , padding with seemingly random bits) can allow for cheating. If you don't trust a man enough to play cards with him, don't play mental cards with him either.

Note: Goldwasser and Micali [GM] have proposed an alternate, more complicated protocol for mental poker, which is evidently more secure.

References

- [Adl] L.M. Adleman, "A subexponential algorithm for the discrete logarithm problem with applications to cryptography," *Proc. 20th IEEE Found. Comp. Sci. Symp.* (1979), 55-60.
- [COS] D. Coppersmith, A.M. Odlyzko and R. Schroepel, "Discrete Logarithms in $GF(p)$," Research Report RC 10985, IBM T.J. Watson Research Center, Yorktown Heights, N.Y., 10598, February 14, 1985.
- [DDHL] R.A. DeMillo, G.I. Davida, D.P. Dobkin, M.A. Harrison and R.J. Lipton, *Applied Cryptology, Cryptographic Protocols, and Computer Security Models*, vol. 29, Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1983. Chapter 4.11, "Compromising Protocols."
- [GM] S. Goldwasser and S. Micali, "Probabilistic Encryption & How To Play Mental Poker Keeping Secret All Partial Information," *Proc. 14th ACM Symposium on Theory of Computing* (1982), 365-377.
- [Lag] J.C. Lagarias, "Knapsack Public Key Cryptosystems and Diophantine Approximation (Extended Abstract)," *Advances in Cryptology, Proceedings of Crypto 83*, (Ed.: D. Chaum), Plenum Press, New York, 1983, 289-301.
- [LLL] A.K. Lenstra, H.W. Lenstra, Jr. and L. Lovasz, "Factoring Polynomials with Rational Coefficients," *Math. Annalen.* 261 (1982), 515-534.
- [PH] S.C. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance," *IEEE Trans. Inform. Theory* IT-24 (1978), 106-110.

- [SRA] A. Shamir, R.L. Rivest and L.M. Adleman, "Mental Poker," MIT/LCS/TM-125, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, February 1979.
- [WM] A.E. Western and J.C.P. Miller, *Tables of Indices and Primitive Roots*, Royal Society Mathematical Tables, vol. 9, Cambridge Univ. Press, 1968.

Security for the DoD Transmission Control Protocol

Whitfield Diffie
Bell-Northern Research
Mountain View, California

1 Introduction

In securing packet switched digital communications, it is possible to add the security measures at almost any layer of the Open Systems Interconnection (OSI) model of network functioning. At one extreme, security may be supplied either by physical protection of the communication links (with no impact at all on network communication protocols) or by independent encryption of the traffic on each link of the network (with little protocol impact). Solutions of this sort are called *link security* and, although widely employed, have the disadvantage of requiring the users to place a high degree of trust in the network. At the other extreme, it is possible, using cryptography, to add security to each individual user level application. This has the advantage of minimizing the user's need to trust the network and thus providing *end-to-end security*, but also has the disadvantage of requiring a multiplicity of implementations.

A natural compromise is to attempt to place the security measures at the lowest point of full end-to-end communications, thereby achieving the benefits of end-to-end security with a single mechanism. As the provider of reliable end-to-end communications, the transport layer is the obvious choice for this location.

In this paper, we will pursue the transport layer approach by examining an existing transport protocol, the U. S. Department of Defense Transmission Control Protocol (TCP), and considering the ways in which this protocol could be made secure.

Our proposals will occur at three levels of compatibility starting with full compatibility with existing TCP and progressing through an upward compatible extension to the possibility of related but incompatible protocols.

2 Overview of TCP

This section provides an overview of the functioning of TCP and is largely drawn or paraphrased from the TCP specification⁴. As in that document, the abbreviation "TCP" will be used to denote both the protocol itself and programs used to implement that protocol.

The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and especially in interconnected systems of such networks. It was explicitly designed for use with the DoD Internet protocol³, but in principle, TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks.

2.1 Facilities

To provide its service on top of a less reliable "network" level communication system requires facilities in the following areas: Data Transfer, Reliability, Flow Control, Multiplexing, and Connection Management.

Data Transfer

The TCP is able to transfer a continuous stream of octets in each direction between its users by packaging some number of octets into segments for transmission through the network. In this stream mode, the TCPs decide when to block and forward data at their own convenience.

The sender can override the asynchronous character of the data transfer by setting the push flag in the TCP send command. This will make the sending TCP transmit all buffered data and set the push flag in the final resulting segment. The receiving TCP on seeing the push flag follows suit by forwarding all buffered data to its user.

TCP also provides a mechanism for communicating to the receiver of data that at some point further along in the data stream than the receiver is currently reading there are urgent data. TCP does not attempt to define specifically what the user should do upon being notified of pending urgent data, but the general notion is that the receiving process should take action to read through the urgent data quickly.

Reliability

Reliability is a complex issue that cannot be completely encompassed within a transport layer protocol. Redundant routing in the network, use of jam resistant communication links, and forward error correction all play a part in a comprehensive program of reliability.

Guarantees of reliability can be divided into two categories of which the second is a necessary building block of the first.

- 1) Assurance that the data will arrive intact.
- 2) Assurance that the receiver will know whether the data have arrived intact or not.

A reliability mechanism providing some elements of each aspect is provided in TCP by the use of sequence numbers and acknowledgments (ACK's) to deliver data undamaged and in order at the destination. Conceptually, each octet of data is assigned a sequence number. The sequence number of the first data octet in a segment is the sequence number transmitted with that segment and is called the segment sequence number. Each segment also carries an acknowledgment number which is the sequence number of the next data octet the receiver expects to arrive. When the TCP transmits a segment, it puts a copy on a retransmission queue and starts a timer; when the acknowledgment for that segment is received, the segment is deleted from the queue. If the acknowledgment is not received before the timer runs out, the segment is retransmitted. At the receiver, the sequence numbers are used to order segments received out of turn and to eliminate duplicates.

TCP's acknowledgment and retransmit mechanism is augmented by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments.

It is important to note that an acknowledgment by TCP does not guarantee that the data have been delivered to the end user, but only that the receiving TCP has taken the responsibility for doing so.

Flow Control

TCP provides a means by which the receiver can govern the amount of information transmitted by the sender. This is achieved by returning a "window" with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received. This window specifies an allowed number of octets that the sender may transmit before receiving further permission.

Multiplexing and Connections

To allow many processes within a single host to use the communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. The concatenation of a port number with the host address from the network communication layer is called a socket.

The reliability and flow control mechanisms require that TCPs maintain certain status information for each data stream. This information, including sockets, sequence numbers, and window sizes, is called a connection and is uniquely specified by the pair of sockets it connects. A connection is defined by a pair of sockets, regardless of the processes plugged in to those sockets and TCP places no restrictions on a particular connection being used over and over again. Each new instance of a connection will be referred to as an incarnation of the connection. A local socket may participate simultaneously in connections to various foreign sockets and all connections are full duplex.

When two processes wish to communicate, their TCP's must first establish a connection (initialize the status information on each side). When communication is complete, the connection is terminated or closed to free the resources for other uses.

The binding of ports to processes is handled independently by each host. However, it is convenient to attach frequently used processes (e.g., a file server or timesharing service) to fixed sockets which are made known to the public. These services can then be accessed through the known addresses. Establishing and learning the port addresses of other processes may involve more dynamic mechanisms in higher protocol layers.

Precedence and Security

In addition to the above features, TCP is also described as providing precedence and security. This, however, is security in the sense of computer operating system security and provides no protection in itself. It is only an option label passed through to the underlying network communication layer, which is expected to operate in a link secure environment. The security label is used by both the ends of the connection and any intermediate nodes to guarantee that classified segments will not be routed either to hosts with inadequate clearance or along paths with inadequate protection.

2.2 The Host Environment

The TCP specification assumes that TCP is a module in a computer operating system and that processes access the TCP much as they would access the file system. The TCP may call on other operating system functions to, for example, manage data structures. The actual interface to the network is assumed to be controlled by a device driver module. The TCP does not call on the network device driver directly, but rather calls on the network level datagram protocol module which may in turn call on the device driver. Despite this assumption the mechanisms of TCP do not preclude implementation of the TCP in a front-end processor, but in such an implementation, a host-to-front-end protocol must provide the functionality to support the type of TCP-user interface described above.

In the environment of a verifiably secure operating system, implementation of TCP within the system itself would be perfectly acceptable from a security viewpoint. In the absence of this

as yet unavailable technology, it is more desirable to isolate TCP together with the cryptographic machinery in a front end computer.

2.3 TCP Interfaces

The TCP/user interface provides for calls made by the user on the TCP to OPEN or CLOSE a connection, to SEND or RECEIVE data, or to obtain STATUS about a connection. These calls are like other calls from user programs on the operating system, for example, the calls to open, read from, and close a file.

The TCP/network layer interface provides calls to send and receive datagrams addressed to TCP modules in hosts anywhere in the internet system. These calls must have parameters for passing the address, type of service, precedence, security, and other control information.

2.4 The Structure of the TCP Segment

Source Port				Destination Port					
Sequence Number									
Acknowledgment Number									
Data Offset	Reserved	URG	ACK	RST	PSH	SYN	FIN	Window	
Checksum				Urgent Pointer					
Options						Padding			
Data									

Figure 2.1 TCP Header Format

The TCP header block carries the sixteen bit names of the source and destination ports, but not the full socket names, which are carried in the underlying network layer datagram. It devotes thirty-two bits each to the sequence number of the first data octet in the segment and, if the ACK bit is set, to the value of the next sequence number the sender of the segment is expecting to receive.

A four bit data offset field specifies the length, in 32-bit words, of the TCP header. Six bits are reserved for possible use in future versions of TCP. Eight control bits explain the segment's purposes:

- URG: Urgent Pointer field significant
- ACK: Acknowledgment field significant
- PSH: Push Function
- RST: Reset the connection
- SYN: Synchronize sequence numbers
- FIN: No more data from sender

The 16-bit window field gives the number of octets beginning with the one acknowledged that the sender is currently willing to accept. The checksum field contains a checksum of the entire segment plus a pseudo-header containing data from the network layer. This checksum was designed for simplicity and makes no attempt to detect intentional tampering. If the URG bit is set, the urgent pointer contains the sequence number of the first octet following the urgent data.

The option field is of variable length and contains any selected options. Each option consists

of either one octet, for a fixed length option, or an option octet, an option length octet, and the option data. Following the options, the header is padded out to an integral number of 32-bit words.

2.5 Establishing and Clearing Connections

Since connections must be established between unreliable hosts and over a potentially unreliable communication network, a handshake mechanism with clock-based sequence numbers is used to avoid erroneous initiation of connections.

A connection, as mentioned earlier, may be opened and closed repeatedly by a variety of different processes. The problem that arises from this is how to identify duplicate segments from previous incarnations of the connection, a problem that is apparent if the connection is being closed and reopened in rapid succession, or if the connection is broken (with loss of memory) and later then reestablished.

A connection is specified in the OPEN call by the local port and foreign socket arguments. In return, the TCP supplies a (short) local connection name by which the user refers to the connection in subsequent calls. There are several things that TCP must remember about a connection and this information is stored in a data structure called a Transmission Control Block (TCB).

The OPEN call specifies whether connection establishment is to be actively pursued, or to be passively attended. A passive OPEN request means that the process wants to accept incoming connection requests rather than attempting to initiate a connection. Often the process requesting a passive OPEN will accept a connection request from any caller. In this case a foreign socket of all zeros is used to denote an unspecified socket.

A connection is initiated by the rendezvous of an arriving segment containing a SYN and a waiting TCB entry created by a user OPEN command. The matching of local and foreign sockets determines when a connection has been initiated. The connection becomes "established" when sequence numbers have been synchronized in both directions.

The procedure used to establish a connection is called a *three-way handshake*. This procedure is normally initiated by one TCP and answered by another. This simplest three-way handshake is shown below. Segment contents are shown in abbreviated form, with sequence number, control flags, and ACK field. Other fields such as window, addresses, lengths, and text have been left out in the interest of clarity.

TCP A		TCP B
1. CLOSED		LISTEN
2. SYN-SENT	--> <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
3. ESTABLISHED	<-- <SEQ=300><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
4. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK>	--> ESTABLISHED
5. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK><DATA>	--> ESTABLISHED

Figure 2.2 Basic 3-Way Handshake for Connection Synchronization

The three way handshake also works if two TCP's initiate communication simultaneously.

TCP A		TCP B
1. CLOSED		CLOSED
2. SYN-SENT	--> <SEQ=100><CTL=SYN>	...
3. SYN-RECEIVED	<-- <SEQ=300><CTL=SYN>	<-- SYN-SENT
4. ...	<SEQ=100><CTL=SYN>	--> SYN-RECEIVED
5. SYN-RECEIVED	--> <SEQ=101><ACK=301><CTL=ACK>	...
6. ESTABLISHED	<-- <SEQ=301><ACK=101><CTL=ACK>	<-- SYN-RECEIVED
7. ...	<SEQ=101><ACK=301><CTL=ACK>	--> ESTABLISHED

Figure 2.3 3-Way Handshake for Simultaneous Connection Synchronization

The examples above do not show connection synchronization using data-carrying segments, but this is perfectly legitimate, so long as the receiving TCP does not deliver any data to the user until it is clear the data are valid (i.e., until the connection reaches the ESTABLISHED state).

The clearing of a connection also involves the exchange of segments, in this case, segments carrying the FIN control flag.

3 Meaning and Scope of Transport Layer Security

In attempting to provide a secure transport layer protocol, we must answer two fundamental questions:

- (1) What does it mean for communications in the transport layer to be secure?
- (2) What does it mean for this security to have been applied by the transport layer?

The answer to the former question, as always, is that transport layer security is the combination of *privacy* (protection against disclosure of message contents to unauthorized parties) and *authentication* (a guarantee that the receiver knows the identity of the sender and that the message has arrived unaltered and without undue delay). In implementing secure transport protocols, however, it is valuable to refine this taxonomy.

As viewed from the transport layer, the opponent in an internetwork environment has the power not only to intercept, record, and examine all data passing over any connection, but to insert or delete messages at will. Privacy protection can be viewed as an attempt to limit the amount of information that the opponent can derive from these activities. Authentication measures are an attempt to assure that the opponents intrusions into the communication channel do not go unnoticed.

In the case of privacy, there is the possibility that even though an opponent is prevented from discovering the contents of any individual message, he is nonetheless able to make valuable deductions from an examination of the timing, length, and distribution of a variety of messages, a technique called *traffic analysis*. Protection for the contents of individual messages is called *message privacy*. Measures that prevent an opponent from studying the overall flow of communications are called *transmission security*.

Authentication is more complicated and is closely tied to the second question. In specifying that the receiver knows the identity of the sender, we must ask in what terms this identity is to be given. A transport protocol provides process to process communication, but these processes are known to the transport layer only through their association with sockets. A guarantee of the identity of the source of a message from the transport layer viewpoint is thus a guarantee that

segments actually emanate from some particular socket. This guarantee will be called protection against *unauthorized connection initiation*.

Given this limited view of the meaning of identity within the transport layer, it is reasonable to ask how socket identity is translated into the identities of entities in which trust is actually vested within the security system. This translation, however, takes place within higher level protocols that make use of the transport layer. The role of the latter is limited to supplying secure socket to socket connections.

The second criterion for the authenticity of data is a guarantee that messages have not been surreptitiously altered during transit; this guarantee is called assurance of *message integrity* or protection against *message stream modification*.

In either of the above cases it is also possible to distinguish different levels of quality in the evidence for authenticity. It is often the case that although the receiver of a message is able to assure himself that he knows the identity of the sender and the message has come through the channel unaltered, he would be unable to establish to a third party that he had not composed the message himself. If the receiver has the means of establishing the identity of the sender to the satisfaction of third parties, we say that the message bears a *digital signature*.

Some intruder actions may take the form, not of altering legitimate messages or even of sending new ones, but of delaying messages either for a limited period of time or indefinitely. The possibility that the intruder will delay messages sufficiently that their meanings have changed is called the threat of *replay*.

When the intruder goes one step further and delays messages indefinitely, the legitimate communicators are said to experience *denial of service*. This threat is often treated differently from others as it is often said that denial of service cannot be prevented, but only detected by authentication measures. A closer examination reveals that this is true of all threats to authentication. An intruder cannot be denied the chance to work mischief on the communication channel, but only prevented from doing it surreptitiously. In the case of message stream modification, however, countermeasures come so directly to the receiver's hand as to becloud the issue: A message that is recognizably inauthentic will be rejected immediately and the intruder will have achieved little. The practical effect of authentication is either to deter the intruder altogether or to convert all attacks into denial of service.

The use of protection against message stream modification opens the question of why false connections must be prevented at all. Data that come from illegitimate connections and data that started out from legitimate connections but were modified *en route* are, after all, indistinguishable to the receiver. Since each message must be authenticated before it is accepted, an unauthorized connection might be opened, but no useful data could be sent over it.

The answer lies in the second question. In saying that security has been supplied by the transport layer, we are saying that the higher level processes that appeal to the transport layer must be placing their faith in it, that the transport layer itself must be operating securely rather than merely serving as the conduit for secure communications. Any authentication procedure required to guarantee segment correctness must therefore be carried out by the TCP's. To limit authentication tests to the data alone and thus allow initiation of a connection (fail to check authenticity of SYN messages) even though no data from that connection would be accepted as authentic, serves only to leave an opening for the opponent to tie up the network with unauthorized connections.

4 Securing TCP Cryptographically

4.1 Cryptography

The basic approach to securing TCP will of course be to encrypt as much as possible of each TCP segment. In so doing, we need make only a few assumptions about the cryptographic system in use. These assumptions describe the operations of which it is capable^{2,6}, including whether it has the public key capability, but say nothing about its strength or internal functioning.

Public Key and Conventional Systems

In using cryptography to provide a secure transport service, either public key or conventional cryptosystems may be used. The advantage of the former are an improvement in the security of key distribution² and the availability of digital signatures. The latter have the advantage, at least for the present, of both higher performance and greater familiarity.

A public key system can perform all of the tasks of a conventional system, even though in some of these it can make no use of its public key capability. A single, public key, cryptosystem might therefore be used for all encryption within a network. At present, however, the low speeds and large block sizes of public key systems make them undesirable for any application in which their special capabilities are not required and a combination of public key and conventional systems is the most satisfactory arrangement.

It is also possible to operate a conventional cryptographic system in the style of a public key system, thereby minimizing the effect on protocol structure of the decision to select one or the other. The user of a public key system employs one key (the other user's public key) for sending messages and another (his own private key) for receiving them. The same approach can be adopted in the conventional case with each user employing one key to encrypt his outgoing messages and another to decrypt the incoming ones.

It is important to remember that a conventional system operating in the public key style does not provide the public key functions; both keys must be treated as secret and no message can be regarded as digitally signed. It is equally important to note that this has little effect within the transport layer. At higher layers the distinction between conventional and public key systems affects the form of the protocols; in the transport layer, it affects only the quality of the protection provided.

Cryptographic systems in the rest of this paper will always be described in the public key form. Each party will have both a sending key (other party's "public key") and a receiving key (his own "private key"). It is convenient for the lengths of keys to be powers of two. Keys for conventional systems are typically between 64 and 256 bits in length while public keys are at present somewhat longer, running from 256 bits up to about a thousand.

Modes of Operation

All cryptographic systems to be used are assumed to be capable of operating in one of the following modes:

- (1) *Cipher block chaining* mode (of which *block mode* is a special case) with blocklength n .
- (2) *Cipher feedback* mode on chunks of text of any size not longer than n .

- (3) Synchronous modes such as *counter driven* mode or *Output Feedback* mode on chunks of text of any size not longer than n .

The most common forms of cipher feedback operate on either a single bit at a time or on eight bits at a time. Because of the octet oriented structure of TCP, eight bit cipher feedback is the most natural choice for encrypting TCP segments. In cipher feedback mode, however, a system can make no use of the public key property and cipher block chaining might therefore be selected for this purpose.

Synchronous modes of cryptographic operation have been popular in communication systems because they do not propagate errors and thus offer good performance in the presence of noise. This feature has no direct effect on TCP itself and is generally less applicable at the transport level of packet switched networks because of error correction at lower levels. Nonetheless, there would be no disadvantage in using synchronous modes with TCP and this might in some cases provide a convenient compatibility with existing equipment.

Message Indicators and Cryptographic Checksums

It is preferable for TCP segments to be independently decryptable, since the alternative requires that sufficient information be left in clear to allow segment ordering before decryption. The cost of this decision is additional information in each segment, telling the receiver the cryptographic state in which to begin decrypting the message. This information is variously called a *message indicator* or *initialization vector* and should, for security's sake, be no less than 64 bits in length.

In both the cipher block chaining and cipher feedback modes of encryption, each item of text is encrypted or decrypted in a manner that depends not only on the key, but on some quantity of the preceeding cipher text. In these modes, the message indicator plays the role of this quantity.

For authentication purposes, the cryptographic system must be capable of generating a cryptographic checksum for each segment transmitted. This checksum is of the order of 64 bits in length and depends on three different types of data:

- (1) Data included in the segment in encrypted form.
- (2) Data included in the segment, but not encrypted.
- (3) Data associated with the segment, but already known to the receiver and not transmitted.

When a public key cryptosystem is employed, the cryptographic checksum can play the role of a digital signature. This can be accomplished either by applying the public key system directly to all of the data to be signed, or by computing a cryptographic checksum with a conventional system and then signing the checksum.

Key Management

Since key distribution is a process that is handled primarily above the transport layer, it will not be examined in detail here. For our purposes, it will be sufficient to assume that when a TCP connection is opened, keys specific to that connection have already been placed in position at its ends.

4.2 *Message Privacy*

Message Privacy is accomplished by encrypting all data in the TCP segment and as much of the header information as possible. The amount of header information that can be protected depends on the degree of compatibility that must be maintained between secure and unsecured TCP. If full compatibility (interoperability with existing TCP implementations) is required, all header data must be left in clear. On the other hand, in a network where all TCPs incorporate security and all segments are required to be encrypted, encryption can be extended to the header as a whole. In a network where both secured and unsecured TCP connections are permitted, some means must be provided for distinguishing between encrypted and unencrypted segments.

Unlike the other elements of the header, the source and destination ports present a particularly difficult problem with respect to encryption. Since connections occur between pairs of ports, port numbers are just the lowest order part of the packet address and from this point of view should merely be passed in clear. This, however, although convenient, is undesirable and probably unnecessary. It is undesirable because the port numbers provide information of great value to a traffic analyst. It is unnecessary since the port numbers (unlike other parts of the address) distinguish between processes all of which are located within a single physically secured location.

If the source and destination ports were encrypted in the connection specific keys, the receiving TCP would have no way of discovering for which of its ports an incoming segment was intended. Its only hope would be to decrypt the segment under the key associated with each possible port. Although procedures of this kind are suitable in some cases, the process would be too time consuming to be applied to each incoming segment.

In order to avoid leaving the port numbers in clear there are two possibilities. Either all segments must be encrypted with a key associated with the host pair rather than the process pair or the port numbers alone must be encrypted with such a key. It appears preferable to encrypt only the port fields using host pair keys, since associating keys solely with host pairs appears to present the same difficulties as having an additional host pair key and is made awkward by the fact that communication is synchronized on a connection basis.

Host pair keys must either be provided by the key distribution mechanism along with the session keys or derived therefrom by the communicating TCP's. Any scheme presents some bookkeeping problems: When a second connection is opened between the same pair of hosts, the corresponding TCP's must cooperate in either maintaining the first host pair key or (probably better, but more difficult) switching to the second. This task cannot be borne by the Key Distribution Center unless there is only one KDC in use by the two hosts and this KDC is required to maintain awareness of all connections in progress.

4.3 *Transmission Security*

Even when the whole segment, including the entire header, is encrypted, the lengths, timings, and host addresses of segments will be visible to traffic analysts. This is a problem that is not readily attacked in transport layer protocols.

The essence of transmission security is concealing traffic patterns from an opponent by sending dummy messages. The role of cryptography in this process is vital but limited: it prevents the opponent from distinguishing real messages from dummies.

Transmission security measures can readily be applied at the link level in circumstances where

the cost of communication does not depend on the volume of traffic. In this case, the link is kept constantly busy with a stream of encrypted data whether there are real messages to send or not.

It is also possible to apply transmission security measures in the network layer of broadcast networks. Under these circumstances, the origins and destinations of messages can be concealed by using cryptography as the addressing mechanism. All messages are encrypted and a station recognizes the messages addressed to it by finding that it can decrypt them successfully. The existence and lengths of messages are harder to conceal. Dummy messages can be sent at little intrinsic cost, but they must be managed very carefully to avoid congesting the network.

Applying transmission security on an end-to-end basis in point to point networks (where addressing information is needed by the intermediate nodes) is extremely difficult and may properly belong to the network layer rather than the transport layer. In order to conceal all traffic flow information, messages must be transported by *flood routing*: The point to point network mimics a broadcast network by routing all messages to all possible addresses. This, of course, is feasible in only the most exceptional circumstances. It may, however, be possible to increase the traffic analyst's burden's, at acceptable costs, by sending only a moderate number of additional messages, particularly if resources are dedicated to relaying messages¹.

4.4 Secure Connection Management

Security gives meaning to the concept of connection above and beyond that already present in TCP. A secure connection is the fundamental service provided by a secure transport protocol and is characterized by the use of a particular set of cryptographic keys. In this light, the TCP concepts of "connection" and "connection instance" deserve further examination.

In TCP a connection is defined entirely by a pair of sockets. Intuitively, this concept is overbroad, failing as it does to distinguish between two quite different cases. In the former, two processes, each of which owns a particular port, repeatedly open and close the TCP connection between them. In the latter, a connection is opened used and closed by a pair of processes, then at a later time the same pair of sockets, hence in TCP terms the same connection, is employed by a new and unrelated pair of processes. For TCP's purposes, these cases are indistinguishable; its concern is solely to be able to discern and reject segments that are not intended for the current incarnation.

We will use the term *session* to distinguish connections of the former type, connections unified by the use of a single set of cryptographic keys. This term reflects the fact that these connections are arranged by key distribution protocols operating in the session layer of network architecture.

It is still necessary to be able to distinguish one incarnation of the connection from the next and so we will further distinguish between *session keys* and *incarnation keys*. The former are supplied by a higher level key distribution mechanism, while the later are arranged locally by the corresponding TCP's in the course of opening the connection.

It is also possible to take the more restrictive view that each incarnation of a connection is a distinct entity and thus to require old keys to be discarded each time a connection is closed and new keys to be distributed each time a connection is opened⁵. We will adopt the viewpoint above as being closer to that of unsecured TCP. Note however, that while a closed unsecured TCP connection leaves no trace in the participating TCP's, a closed secure connection requires a key to be preserved for later use either by TCP or by some closely associated mechanism.

In many applications a secure connection will be limited to a single incarnation. The procedure in this case, however, is the same as for the multi-incarnation case: an incarnation key is derived from the connection key during the open.

Establishing a Secure Connection

Secure connection initiation requires the addition of a challenge and response authentication procedure to the three way handshake. Before connection initiation can begin, keys must have been delivered to the corresponding TCP's. In the fundamental case, where one process is initiating a connection to another, the key distribution mechanism will also provide each TCP with a specification of both sockets. Once this has occurred, the "calling" TCP begins an active open sequence and the "called" TCP begins a passive open. The basic structure of such an exchange follows; the issue of how the data are incorporated in TCP segments depends on the degree of compatibility with unsecured TCP that is required and is discussed in later sections. As described earlier, all keys will be presented in pairs in the public key form.

The calling TCP, A, constructs a SYN segment to which a challenge has been added.

TCP-A→TCP-B: $\{A's\ challenge\}^{B's\ public\ key}$

When the called TCP, B, receives this segment it returns a SYN packet to A both answering A's challenge and presenting a challenge of its own.

TCP-B→TCP-A: $\left\{ B's\ response = \{A's\ challenge\}^{B's\ private\ key}, B's\ challenge \right\}^{A's\ public\ key}$

Using TCP B's public key, TCP A can decrypt the first half of this message to verify that its challenge has been answered correctly and can recognize the latter part as a challenge to which it must respond.

TCP-A→TCP-B: $\{B's\ challenge\}^{A's\ private\ key}$

This exchange, in addition to unsecured TCP's synchronization of sequence numbers, demonstrates to each party that the other is the party to which it had been referred by the key distribution mechanism. At the same time it serves to exchange pieces of information (the challenges) that will serve as the keys for the current incarnation.

The requirement in unsecured TCP that sequence numbers be generated in a non-repeating, clock dependent, manner is replaced by a similar but more exacting mechanism for generating the incarnation key. This allows the sequence numbers themselves always to begin at zero, since segments from previous incarnations can be recognized as being encrypted under outdated incarnation keys.

TCP allows the possibility that both ends of a connection may attempt to open simultaneously. Although this is unlikely at the beginning of a secure connection, it can occur when a connection is reincarnated. The only effect of independent opens is that the second message above will appear as two distinct messages rather than one combined message.

Passive Open with Unspecified Foreign Socket

The secure connections discussed above all take place between fully specified socket pairs; unsecured TCP, however, allows the possibility of a passive open with an unspecified foreign

socket. Among secure TCPs, this event can only occur in limited cases, but these cases are very important. This mechanism both impinges on the domain of key distribution and is required by the key distribution center.

When a listening TCP receives an encrypted SYN from an unpredetermined foreign socket, it must determine what key to use. This problem is dramatically simplified if the connection keys are public keys, in which case the session key will be a public key for the listening process regardless of the identity of the calling process. Using conventional keys, however, this determination is more complex.

If the calling process is always assigned the same port by its local TCP, then the listening TCP can determine the correct key from the foreign socket. This is probably the most general possibility that can be allowed since in any other case the contacts, although between the same two processes, are not in the TCP sense the same connection.

Message Integrity

Message integrity is gained by adding a cryptographic checksum (also about 64 bits in length) to the segment. This checksum covers the pseudo-header, header, and data and must be correct in order for a segment to be acknowledged.

This mechanism also extends the sender identification established during connection initiation by demonstrating that the sender of the segment knows the incarnation key that was agreed on during the challenge and response.

4.5 Detection of Replay

Unsecured TCP provides a mechanism for recognizing and rejecting segments from previous incarnations that have been lost long enough in the network to be mistaken for segments from the current incarnation. Secure TCP must reject in addition segments held for arbitrarily long periods, and subsequently replayed, by an opponent. Fortunately, the cryptographic techniques used to provide security also provide a simpler and more reliable means of making this distinction.

There are two fundamental means for judging the timeliness of messages. If the sender and receiver have synchronized clocks, a message whose integrity is guaranteed can also be authenticated as timely by examination of an included time field. This time field can be expressed either in hours minutes and seconds or, as with TCP's sequence numbers, in terms of the amount of data sent and received. If synchronized clocks are not available timeliness can be verified by a challenge and response procedure. Data will be recognized as current if they are tied to the response to a current challenge.

Secured TCP uses both of these mechanisms. The incorporation of a challenge and response procedure in initiating the connection guarantees the timeliness of the SYN segments. These in turn serve to synchronize a clock (the sequence numbers) in terms of which the timeliness of all later segments is verified.

Replayed segments will be detected as inauthentic either because they come from earlier incarnations of the connection, and are thus encrypted in the wrong incarnation key, or because they come from earlier in the same incarnation, and thus have the wrong sequence number.

4.6 Detecting Denial of Service

Some protection against denial of service is built in to TCP through the acknowledgment mechanism: a sending TCP cannot remain unaware that a segment has failed to reach its destination. A TCP that is not transmitting, however, but merely waiting for a message from the other end of the connection has no way of knowing if this message has been blocked or merely has yet to be transmitted.

To counter this possibility, a TCP that has not received a segment in some time can challenge the other end of the connection to demonstrate its availability and authenticity. This exchange is similar to that used to initiate an incarnation and can be used to change the incarnation key at unpredictable times during the session. This procedure can be used not only to detect denial of service, but to counter subtle vulnerabilities that make the use of a public key system to exchange conventional incarnation keys less secure than the use of public keys throughout².

5 Full Compatibility—An Added Layer of Protocol

Full compatibility with TCP does not permit encryption of the header or even any changes or additions thereto. Any action that is to be taken must consist solely of additions to and encryption of the user data. These additions, furthermore, must take place prior to, and therefore in ignorance of the actions of TCP. This renders such otherwise plausible acts as adding a cryptographic checksum of the entire TCP segment infeasible.

The effect is of the addition of an extra layer of protocol at the upper edge of the transport layer. A host adopting this approach will present to the world an entirely correct TCP appearance and may even have unsecured conversations with standard TCPs. A process requiring a secure connection, however, must make use of TCP not directly but through the added security layer.

Under these circumstances some of TCP's functions are difficult or impossible to duplicate without building almost full transport layer functioning into the added security layer.

5.1 Data Privacy and Authentication

From TCP's point of view, the security layer is a user process and TCP will therefore hand it data that TCP believes to be damage free and in proper order. This allows the security layer to protect the privacy of the data by encrypting them as a single *cipher chain* and frees the security layer from the need to add a message indicator to each segment. The result is a reduction in overhead when transmitting ordinary data.

One price that is paid for this reduction in the normal case is increased overhead in transmitting urgent data. When the security layer hands TCP a buffer with the urgent flag set, it must incorporate a message indicator as the first element. When the security layer receives urgent data from TCP, it must treat the first portion as a message indicator and decrypt the data accordingly.

For authentication, the security layer must also compute cryptographic checksums on both the data and certain information from the TCP header and pseudo header. These are added by the sending security layer and used by the receiving one to test for alteration. In order to verify timeliness, these checksums must cover some equivalent of the sequence number; in order to detect segments maliciously routed back to the sender, they may need to cover the full socket pair.

Fortunately, although the elements of the header generated by TCP are not available to the security layer, the information given to TCP in the OPEN or SEND calls is. Requests to open or

use a secure connection are made to the security layer and passed thereby to TCP; this gives the security layer access to both socket addresses, although not the sequence number, acknowledgment, or window. To replace the sequence number to which it lacks access, the security layer generates a sequence number of its own by counting octets.

For the sake of generality, it is desirable that authentication be accomplished without imposing any structure on the data beyond the segmentation carried out by TCP itself. This, however, presents a difficult problem. On transmission, the cryptographic checksums can be attached to the data given to TCP by the security layer. Recognizing these checksums in the received data is another matter. The security layer is not only unable to get TCP header information, but on receiving, it is unable to distinguish the data comprising individual segments.

In order to make the segment structure of received segments visible to the security layer markers must be placed in the data stream where they will be passed through by TCP. Since TCP provides a transparent channel, any code reserved for this purpose can be expected to make occasional independent appearances in the data. Preventing such patterns from causing disruption requires use of bit or character stuffing techniques to alter the reserved pattern whenever it is seen by the sending security layer.

In the event that inauthentic data are detected by the receiving security layer, its options are quite limited. It has no meaningful way to reject the segment, which has already been accepted by TCP, unless it duplicates the entire acknowledgment and retransmission mechanism. On the other hand, data that have already been accepted by TCP yet are found to be inauthentic must have been intentionally manipulated and the security layer can reasonably respond by sending an alarm message back to the user process and a reset to the TCP connection.

5.2 *Connection Initiation*

Use of a separate security layer means that the triple handshake of TCP must be completed and then mirrored in a similar cryptographic handshaking procedure by the security layer. At first glance, it would appear that these two processes could be at least partially combined by making use of the data carrying abilities of TCP SYN segments. This approach fails, however, because TCP will not deliver the data in these segments to its user (the security layer) until its own connection setup process is complete.

5.3 *Detecting Denial of Service*

The separation of the security layer from TCP deprives the former of the level of denial of service protection supplied by TCP. The security layer can check arriving data for integrity, but would require a full acknowledgment mechanism of its own to be sure that data it sent had arrived. Implementation of more refined denial of service detection, however, is straightforward and mimics the initial authentication exchange.

6 *An Upward Compatible Extension of TCP*

An upward compatible extension of TCP simplifies the introduction of security by allowing the security mechanism direct access to the structure of TCP segments. This permits the segment as a whole, including almost all of the header information to be encrypted. The exceptions are a bit indicating whether or not the segment is encrypted and perhaps the local and foreign port addresses.

The problem of encrypting the port addresses has been touched on in an earlier section. They

cannot be encrypted in a connection specific key since they are used precisely to distinguish between the various possible connections. They can, however, be encrypted in a host pair key. This has little effect on the principal functions of TCP and presents primarily a problem of installing and changing these keys at suitable times.

In most cases, the receiving TCP does not need to distinguish between encrypted and unencrypted segments because it will have been informed by the key distribution mechanism that an encrypted connection is to be created between specified local and foreign sockets. Aside from the possibility that encrypted and unencrypted segments may be allowed in the same session, the principal circumstance in which TCP might be required to distinguish is that of a passive open with unspecified foreign socket.

Since the receiving TCP must be able to read the "encrypted" bit before decrypting the segment, this bit must be located in a fixed position relative to the segment's beginning, a constraint that precludes the use of the option area. The best solution appears to be using one of the reserved bits to indicate an encrypted segment. This assumes that the receiving TCP has only one cryptographic system at its disposal or that it has been informed by some other means of which system to use. This, however, is a natural assumption, since any other approach would be open to criticism on transmission security grounds.

Encrypting the whole segment requires that decryption be carried out before segment reordering and therefore that each segment must be independently decryptable; this in turn mandates the addition of a message indicator to each segment. Since the receiving TCP cannot decrypt the segment correctly unless it is able to locate the message indicator, this item, like the "encrypted" bit must occur in a fixed position, even though it only occurs in encrypted segments.

One more item must be added to the segment: a cryptographic checksum. Unlike the message indicator, this need not be locatable prior to decrypting the segment. This allows greater freedom in its placement, but it seems clean and convenient to put it directly after the message indicator.

Source Port				Destination Port			
Sequence Number							
Acknowledgment Number							
Data Offset	Reserved	E	U	A	P	R	S
		N	R	C	S	S	I
		C	G	K	H	T	N
				Window			
Checksum				Urgent Pointer			
Message Indicator							
Cryptographic Checksum							
Options						Padding	
Data							

Figure 6.4 Encrypted Segment Header

The checksum occurs in every encrypted segment and is calculated from the entirety of the pseudo header, the header, and the data with the exception of the checksum itself, which may either be omitted from the calculation or replaced by zeros. In encrypted segments, the function of the 16 bit, non-cryptographic checksum in determining segment acceptability is supplanted by use of the cryptographic checksum.

In segments with the SYN flag set, additional data must be incorporated for the challenge and response aspect of the handshaking. These are probably best included in option fields, and we will add the two options CHAL and RESP. This permits these options to be used by themselves for various reinitializations of the incarnation key.

6.1 *Privacy, Reliability, and Authenticity*

Except during the opening of a connection, TCP's many functions operate protected, but barely affected, by encryption. An opponent examining intercepted segments can observe that their encrypted bits are on, and can confirm this observation by attempting to read the data, but can observe nothing more than the total length of the segment. The connection to which the segment belongs, the segment's data, and the various fields that would reveal how many octets have been sent, how many acknowledged, and whether the segment is a SYN, FIN, or retransmission are all concealed.

The acknowledgment and retransmission mechanisms of secured and unsecured TCP operate in exactly the same way except that the cryptographic checksum replaces the non-cryptographic in deciding whether to accept a segment. An opponent can neither alter the subscribers' data nor affect the connection's behavior by inserting phony control segments, since any segment received is first judged for authenticity by its checksum and then for timeliness by its sequence number.

6.2 *Secure Connection Initiation*

The unsecured three way handshake assures both participating TCP's of the timeliness of the connection and allows them to reject stray segments from previous incarnations with high reliability. Each side chooses an initial sequence number for the current incarnation, sends this sequence number to the other TCP and receives in return an acknowledgment of this choice. Each TCP must both have acknowledged the other's starting sequence number and received an acknowledgment of its own before it will regard the connection as open and accept data for passage to its user.

The initiation of secure TCP connections follows this pattern in form and extends it in objectives, verifying the timeliness of the connection as well as the more fundamental fact that the participating parties share a compatible set of cryptographic keys.

Rather than agreeing on initial sequence numbers, secure TCP's agree on a set of keys for use in the current incarnation. This allows segments from previous incarnations to be detected not on the basis of bad sequence numbers, but on the inability of the receiver to decrypt them and derive a correct cryptographic checksum. This procedure is less prone to accidental failure than the unsecured version since keys are never less than twice as long as TCP's 32-bit sequence numbers and accidental repetitions are correspondingly less likely. It also frees the participants from the need to select random starting points in the sequence number space and allows both to begin at zero.

In a secure connection, sequence numbers can never be permitted to cycle during the use of a single key since this would not allow new segments to be distinguished from old (played back) segments with the same sequence number. In fact, keys are not expected to remain in use for nearly this long, but rather are changed periodically by a mechanism to be discussed in connection with detecting denial of service.

In the most common case of secure connection initiation, one end of the connection, which we

will denote as A, starts proceedings with an active OPEN while the other, B, makes itself receptive to an arriving segment by doing a passive OPEN.

A→B:	SEQ = 0		CTL = ENC, SYN		} B's public key
	Message Indicator				
	Cryptographic Checksum				
	A's challenge				

B responds by answering A's challenge and posing one of its own:

B→A:	SEQ = 0	ACK = 1	CTL = ENC, SYN, ACK	} <i>A's public key</i>
	Message Indicator			
	Cryptographic Checksum			
	B's challenge			
	<i>B's response</i> = { <i>A's challenge</i> } ^{<i>B's private key</i>}			

A is now able to verify that B has correctly answered its challenge by decrypting B's answer with B's public key. The cryptographic version of the three way handshake is concluded by A's answer to B's challenge:

A→B:	<table> <tr> <td>SEQ = 1</td> <td>ACK = 1</td> <td>CTL = ENC, ACK</td> </tr> </table>			SEQ = 1	ACK = 1	CTL = ENC, ACK	<i>B's public key</i>
	SEQ = 1	ACK = 1	CTL = ENC, ACK				
	Message Indicator						
	Cryptographic Checksum						
	<i>A's response</i> = { <i>B's challenge</i> } <i>A's private key</i>						

Once B has checked A's signature on the challenge, both processes are in an ESTABLISHED state and are willing to accept and acknowledge data. As with an unsecured handshake, the synchronization messages can carry data, but these data must not be accepted and passed on to the user until the handshake is complete.

Throughout connection initiation, the segments exchanged are encrypted in the correspondents' session keys. Once the TCP's enter the ESTABLISHED state, however, they will switch to using an incarnation key, manufactured from the exchanged challenges, for the duration of the incarnation. There are various ways to produce such an incarnation key, but we will adopt the convention that the challenges are treated as exchanged public keys, regardless of whether a public key or conventional system is in use. Each side of the connection will thus transmit in one key and receive in another.

The switch from the session keys to the incarnation keys opens the possibility of doubt on the receiving TCP's part about which key to use in decrypting an incoming segment. Once an incarnation key has been selected, this will become the key of choice and most segments otherwise encrypted will represent errors. If, however, a segment fails to decrypt correctly using the incarnation key, the session key can be tried.

6.3 Detecting Denial of Service

Denial of service can be reliably detected by the sender of messages, say A, through its failure to receive acknowledgments. After a number of attempts that will vary with circumstances, it will

respond by sending a trouble report to its user. An intruder cannot defeat this strategy by sending phony acknowledgments because he is unable to make his phonies cryptographically acceptable.

The prospective receiver, B, on the other hand has no way of knowing that segments intended for him are being prevented from reaching their goal. If B is to discover this, he must send messages to A that will provoke a response.

The same challenge and response mechanism used in establishing the connection is suitable for this purpose. Either party may at any time during the connection, whether it feels deprived of incoming data or not, send a new "public key" to the other party and expect a satisfactory response. This challenge can be given a segment to itself or combined with user data. This latter possibility helps to prevent an opponent from distinguishing such messages from data and allowing only the former to pass.

It is interesting to note that challenges posed in this manner arrive in segments encrypted with the current incarnation key, but must be signed with the receivers private key. A correct response therefore guarantees that the responder knows both.

After a key change, arrival of legitimate data encrypted with the old key is not unlikely and the receiver must be prepared to hold it until all segments sent before the key change have been received and acknowledged. The sender is under no such obligation and can freely retransmit unacknowledged segments in the new key rather than the old.

As noted earlier, this mechanism not only serves to detect denial of service but to prevent recycling of sequence numbers.

7 Incompatible, But Related, Protocols

TCP is best suited to establishing connections across which substantial amounts of data will be transferred asynchronously in both directions; it is inefficient for transmission of small amounts of data such as remote procedure calls and, due to its insistence that every segment must be received undamaged and acknowledged, ill suited to carrying real time data such as voice.

TCP's inefficiency in transmitting small amounts of data has a direct effect on its suitability for communication between a KDC and its clients since key exchange requires only very short messages and, unless each client maintains a constant connection with the KDC, TCP will add substantial overhead. The specialization of TCP has a more profound effect, however, on its utility as the common denominator of secure communications and the primary location for the network security mechanism. This utility is dependent on the assumption that all processes above the transport layer will make use of TCP and can thus relay on it to provide security. If instead, there must be several secure transport layer protocols, not only must security be incorporated in all of them, but each must be provided with access to cryptographic hardware.

TCP segments are best viewed as "programs that are mostly data" and TCP as an interpreter for executing these programs. In this view, an effort to make TCP more flexible would probably change it from a language with a nearly fixed length instruction set to one with a variable length instruction set. Instead of requiring that all of the various fields: acknowledgment number, checksum, window, etc. be present in every segment, the segment header would begin with the control bits and incorporate additional fields as needed.

E N C	U R C G	A P R H	P S T	R S T N	S Y N	F I N	E C K	Reserved				Data Offset
Source Port								Destination Port				
Sequence Number												
Acknowledgment, Window, etc. (as needed)												
Options and Padding												
Data												

Figure 7.5 Possible Alternative Header Format

In a configuration analogous to that presented in the previous section, a security control bit would be set. This would indicate that the segment must be decrypted and that the acceptance test would be cryptographic. In this case a checksum control bit would not be present and no non-cryptographic checksum would be performed. The present TCP layout in these respects would be modeled by turning the secure bit off and the checksum bit on. In a local area network that was considered to be both reliable and secure neither bit might be set.

Acknowledgement

I am grateful to Steve Kent for his little known technical note, "TCP and Communication Security," which provides an insightful exploration of the problems encountered in producing a secure version of TCP.

References

- [1] D. L. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, Vol. 24, No. 2, pp. 84-88, February 1981.
- [2] Whitfield Diffie, "Conventional Versus Public Key Cryptosystems," in *Secure Communications and Asymmetric Cryptosystems*, Edited by Gustavus J. Simmons, Westview Press, Boulder, Colorado, 1982.
- [3] "DoD Standard, Internet Protocol," Information Sciences Institute, University of Southern California, Marina del Rey, California, RFC 791, September 1981.
- [4] "DoD Standard, Transmission Control Protocol," Information Sciences Institute, University of Southern California, Marina del Rey, California, RFC 793, September 1981.
- [5] Steven T. Kent, "Some Thoughts on TCP and Communication Security," MIT, Laboratory for Computer Science, Local Network Note, No. 6, 4 May 1977.
- [6] "Modes of Operation for the Data Encryption Standard," National Bureau of Standards, Federal Information Processing Standards Publication 81, 1980.

Symmetric Public-Key Encryption

Zvi Galil^{1, 2, 3} Stuart Haber^{1, 3} Moti Yung^{1, 3, 4}

¹ Department of Computer Science, Columbia University

² Department of Computer Science, Tel Aviv University

Summary

Public-key encryption would seem to be inherently asymmetric, in that only messages sent to a user can be encrypted using his public key. We demonstrate that the use of interactive protocols for sending encrypted messages enables a **symmetric** use of public keys; we give cryptographic protocols for the following tasks:

1. Probabilistic encryption, using the same public key, both of messages that are sent to a particular user as well as of messages that the user sends to others, without compromising the key. We propose a public-key cryptosystem based on these protocols which has only one key, owned by a cryptographic server.
2. Authentication both of the sender and of the receiver of a probabilistically encrypted message.
3. Probabilistic encryption which is provably secure against both chosen-message and chosen-ciphertext attack.

December 1985

³ Supported in part by NSF grants MCS-8303139 and DCR-8511713.

⁴ Supported in part by an IBM graduate fellowship.

1. Introduction

As introduced by Diffie and Hellman and further studied by many authors, public-key encryption would seem to be inherently asymmetric: messages sent to user A are encrypted using A's public key [6, 16]. This is true both for deterministic [15, 13] and for probabilistic [8, 3, 5] implementations of the Diffie-Hellman model.

In this paper we suggest that users follow an interactive protocol in order to send probabilistically encoded messages, and show how this allows the symmetric use of public keys. A's public key will be used to encode messages that are sent to A as well as to encode messages that A sends to others, without compromising the key. We contrast our protocol with previous interactive schemes, in which public-key encryption was used in order to distribute additional private keys that could be used symmetrically by pairs of users [9, 18]; our scheme enables symmetric use of the public key itself.

This capability is useful in a number of cryptographic settings. For example, it enables a casual user who is not registered in the central file of public keys to receive a private message. It can also be used in a cryptographic network with a trusted central server, through which all messages are routed; here only a single public key is needed (cf. [12]).

We extend our scheme so as to enable the symmetric authentication of an encoded message --- that is, the authentication both of the sender and of the receiver of the message. This is the first such scheme, in the setting of probabilistic encryption, that uses only the encryption keys.

Probabilistic encryption was proposed in order to hide from an eavesdropper all partial information about an encoded message. However, all of the systems discussed in the literature are vulnerable to chosen-ciphertext attack. We give a refinement of our protocol (based on [11]) which is provably secure against chosen-ciphertext attack. In addition, we give another symmetric public-key encryption scheme, this one based on a minimum-knowledge interactive proof-system, which is also chosen-ciphertext secure [7].

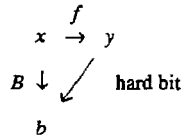
2. Background

In the model introduced by Diffie and Hellman, each user A in a public-key crypto-system has a public encryption algorithm E and a private decryption algorithm D . Any other user encrypts a message M that he wishes to send to A by computing the ciphertext $E(M)$; only A is capable of computing $D(E(M)) = M$ to recover the original message [15, 13]. In order that the ciphertext reveal no partial information about the message, it has been suggested that E and D be probabilistic algorithms [8, 3, 5].

We would like A to be able to use her own public key in order to send an encrypted message to another user B. In order to do this securely, so that no other users can decrypt the message, it seems necessary to make the transfer of a message depend on an interactive protocol between A and B. In this way B can help to choose the random input to the probabilistic encryption and decryption algorithms. In the next section we will show how to implement this idea; first we sketch the methods of probabilistic encryption that we will use.

The security of the protocols that we discuss in this paper relies on the existence of *hard bits*, that is, Boolean predicates B for which there is an efficient reduction to B of an assumedly intractable number-theoretic problem.

Specifically, we will assume that we are given functions of the following form. Let $D \subseteq \{0, 1\}^n$ be a (non-sparse) set of n -bit strings, and suppose that $f: D \rightarrow D$ is a *one-way trapdoor permutation*. Suppose in addition that $B: D \rightarrow \{0, 1\}$ is an (efficiently computable) Boolean predicate such that f^{-1} is efficiently reducible to the "hard bit" $B \circ f^{-1}$.* (Yao has shown that, even without such a predicate, f can be used to generate pseudo-random bits [17].)



Such a function and its associated Boolean predicate may be used as a *cryptographically strong generator* of pseudo-random bits. Given any element $x \in D$, and two integers $j \leq k$, we will define

$$G(x, j, k)$$

to be the bit-sequence

$$B(f^j(x)), B(f^{j+1}(x)), \dots, B(f^k(x)).$$

$$\begin{array}{ccccccc}
 x & \rightarrow & f(x) & \rightarrow & f^2(x) & \rightarrow & \dots \rightarrow f^n(x) \\
 & & \downarrow & & \downarrow & & \downarrow \\
 \text{pad}(x) = & & b_1 & & b_2 & & \dots & b_n
 \end{array}$$

If elements $x \in D$ are chosen at random, then the bit-sequences $\text{pad}(x) = G(x, 1, n)$ are indistinguishable (in time polynomial in n) from truly random bit-sequences. That is, an efficient algorithm which could distinguish between the two sorts of sequences with non-negligible probability could in turn be converted to an efficient algorithm for computing f^{-1} , contradicting the assumption that f^{-1} is hard. (For a more complete account of this, see [4, 17, 2].) The provable security of these bit sequences permits us to use them to simulate one-time pads.

The schema just described is an abstraction of two different methods of pseudo-random bit-generation. That of Goldwasser, Micali, and Tong [9] requires an n -bit integer $N = pq$ which is the product of two primes satisfying either $p \equiv q \equiv 3 \pmod{8}$, or $p \equiv q \equiv 7 \pmod{8}$. The domain D is the set $\{x \in \mathbb{Z}_N^* \mid 0 < x < N/2, (\frac{x}{N}) = 1\}$; we define the function f by $f(x) = \pm x^2 \pmod{N}$, choosing either $+$ or $-$ so that $0 < f(x) < N/2$, and we define the predicate $B: D \rightarrow \{0, 1\}$ by $B(x) = \text{parity}(x)$. Both f and B can be efficiently computed. The trapdoor information for f is the factorization of N ; this information enables the efficient computation of $(f^{-1})^j$ [5]. The security of the hard bit of this scheme was proved by Alexi, Chor, Goldreich, and Schnorr [1].

*We may also have $B: D \rightarrow \{0, 1\}^k$, where B is a k -bit "predicate" all of whose bits are *simultaneously* hard. (In this case, the cryptographic applications -- modified in the obvious manner -- are more efficient by a factor of k .) The proofs go through with little change.

The pseudo-random bit-generator of Blum, Blum, and Shub also involves squaring modulo a large integer N [3].

The first probabilistic encryption scheme, due to Goldwasser and Micali [8], does not use a pseudo-random bit-generator; instead, each bit of a message is encoded either as a random quadratic residue or as a random quadratic non-residue modulo a large integer N which is the product of two primes. More precisely, a public key in this scheme consists of N together with y , a quadratic non-residue mod N with Jacobi symbol $+1$. To encode a bit, one chooses $x \in \mathbb{Z}_N^*$ at random and sends either $x^2 \bmod N$ (a random residue) or $yx^2 \bmod N$ (a random non-residue), according to whether the bit is 0 or 1. For any bit-string s , we will use $E_{N,y}(s)$ to denote the set of possible encodings of s ; thus, if s has length l then $E_{N,y}(s)$ consists of l -tuples of residues and non-residues mod N . Gaining any partial information about an encoded message is as hard as distinguishing residues from non-residues mod N , which appears to be an intractable problem without knowing the factorization of N . The trapdoor information which enables efficient decoding is exactly this factorization.

3. Symmetric Encryption Scheme

We describe here how to encode and decode, using a cryptographically strong bit-generator constructed with a trapdoor function as described above.

Let f be the trapdoor function whose specification is contained in user A's public file; that is, all users in the network can compute f quickly, but only A has the trapdoor information enabling her to compute f^{-1} . Generalizing the scheme of Blum and Goldwasser [5], we show how any other user B can send an encrypted message to A using f ; we then describe how --- using the same function f --- A can send a securely encrypted message to B.

In both cases, the cleartext being sent is an l -bit message M (where l is polynomial in n .) For any element $x \in D$, we will use the notation $\text{pad}(x) = G(x, 1, l)$.

Protocol 1

In the 'forward' or usual direction, B chooses an element $x \in D$ at random, and computes $\text{pad}(x)$, $C = \text{pad}(x) \oplus M$, and $X = f^{l+1}(x)$. B sends to A the encryption of M , namely the pair $[C, X]$. A can decode by computing $x = f^{-(l+1)}(X)$ and $C \oplus \text{pad}(x) = M$.

Encrypting messages in the opposite direction seems to require some additional communication. We propose the following protocol for A to send the message M to B.

Protocol 2

- A \rightarrow B: "Hi"
- B chooses x at random in D ,
and computes $\text{pad}(x)$ and $X = f^{l+1}(x)$
- B \rightarrow A: X
- A computes $x = f^{-(l+1)}(X)$, $\text{pad}(x)$,
and $C = \text{pad}(x) \oplus M$
- A \rightarrow B: C

- B computes $C \oplus \text{pad}(x) = M$

Notice that the information that is available to the eavesdropping adversary, namely X (which serves as an encoding of the seed x) and C , is the same in both protocols.

Under known-message attack, the present scheme is as secure as the problem of inverting f . To be more precise, assume that an eavesdropper witnesses polynomially many executions of one or both of the above protocols (polynomial in n), and that he knows the message M_i that was sent in the i th execution (either sent by user B_i to A, following Protocol 1, or sent by A to user B_i , following Protocol 2). Suppose that, after some polynomially bounded computation, the eavesdropper is able either (a) to correctly simulate the behavior of A in order to send a message of his (the eavesdropper's) choice; or (b) to decode (with probability non-negligibly better than $1/2$) one bit of the next message that A sends or receives. Then our eavesdropper must be able to compute f^{-1} quickly. The proof relies on the (polynomial-time) indistinguishability of the pseudo-random bits of $\text{pad}(x)$ from truly random bits.

We now show how the bit-by-bit probabilistic encryption scheme of Goldwasser and Micali can also be adapted so as to encode messages either to or from the owner of a public key. Let user A have the public key (N, y) .

As in the original scheme, user B can send a message M to A by probabilistically computing an element $e \in E_{N,y}(M)$ and sending it to A. Knowing the trapdoor information, A can recover M from e . We will call this Protocol 1*.

In order for A to send an l -bit message M to B, the two users execute the following protocol.

Protocol 2*

- A \rightarrow B: "Hi"
- B chooses $p \in \{0, 1\}^l$ at random, and probabilistically computes $e \in E_{N,y}(p)$
- B \rightarrow A: e
- A computes p and $C = p \oplus M$
- A \rightarrow B: C
- B computes $C \oplus p = M$

As before, Protocol 2* is as secure as encryption in the usual direction.

3.1. Applications

An immediate application of these protocols is to allow encoded messages to be sent to casual users in a network without requiring that they undergo any special procedure such as having a key registered in a public-key library.

If a network includes a trusted central server, then any message can be sent via the server, encoded using the server's public key; when A wants to send a message to B, she sends it to the server using Protocol 1, and the server

sends it on to B using Protocol 2. In this case, only a single public key is needed; there is no need to initialize and maintain a public-key library. This may increase the security of the system and decrease its cost.

We observe that if this suggestion is implemented using bit-by-bit probabilistic encryption, i.e. with Protocol 1* and Protocol 2*, then it is easy to prove that an adversary gains no advantage whatsoever from over-hearing (what he knows to be) two encryptions of the same message.

4. Authentication

What has been presented so far is purely an encryption scheme. We now assume that each user has his own public key, and we consider the problem of user authentication.

We distinguish two authentication problems that arise when A sends a message to B. There is the problem of *sender authentication*, which is to convince B that it was indeed A who sent the message, and the complementary problem of *receiver authentication*, which is to convince A that it is indeed B who received the message.

In the usual public-key encryption scheme, in which messages to user A are encrypted using A's public key, receiver authentication is assured by the fact that only A knows the private key which is necessary for decryption; on the other hand, there is no automatic provision for authenticating the sender of a message. A malicious user C can send a message to A, claiming to be B.

In a standard deterministic public-key scheme, the usual modification to assure sender authentication is by means of a "digital signature", specifying that the message be further encrypted using the sender's private key; that is, B encrypts his message M to A as $D_B E_A(M)$ [6]. On the other hand, of the several proposed probabilistic public-key schemes, none seems to allow for an easy modification that assures sender authentication.

In our scheme, whereby A sends messages to others using her own public key, the authentication problems are correspondingly reversed. Sender authentication is guaranteed by the fact that only A knows the trapdoor information (for inverting f) which is necessary for encryption, while receiver authentication is no longer assured. That is, a malicious user C can masquerade as B in an execution of Protocol 2, since the only secret information in the scheme is that of A.

We suggest that a simple modification of the protocols presented in the last section allows A to send M to B so that both A and B can authenticate their identities to each other. Let f_A, f_B denote the public encryption functions for users A and B, and let D_A, D_B be their domains. In the following protocol for A to send an authenticated l -bit message M to B, $\text{pad}(x)$ and $\text{pad}(y)$ are computed using f_A and f_B , respectively.

Protocol 3

- $A \rightarrow B$: "Hi, this is A sending a message to B."
- B chooses x at random in D_A ,
and computes $\text{pad}(x)$ and $X = f_A^{l+1}(x)$.
- $B \rightarrow A$: X
- A computes $x = f_A^{-l+1}(X)$, $\text{pad}(x)$,

chooses y at random in D_B .

computes $\text{pad}(y)$, $Y = f_B^{l+1}(y)$, and $C = M \oplus \text{pad}(x) \oplus \text{pad}(y)$

- $A \rightarrow B$: $[C, Y]$
- B computes $y = f_B^{-l+1}(Y)$, $\text{pad}(y)$,
and $C \oplus \text{pad}(x) \oplus \text{pad}(y) = M$

As with Protocols 1 and 2, this protocol is secure against known-message attack. Moreover, impersonating A or B is as hard as inverting f_A or f_B , respectively.

5. Chosen-Ciphertext Security

In a chosen-ciphertext attack, the adversary is allowed to have a ciphertext of his choice decrypted. Several proposed cryptosystems which are secure against weaker sorts of cryptanalytic attack are easily seen to be vulnerable to the chosen-ciphertext attack. In this section we show how introducing interaction to the cryptosystem enables us, for the first time, to achieve provable security against this attack.

For an example of the chosen-ciphertext attack, consider Protocol 2 specified above. B, the receiver of the message, can cheat in the following way. Instead of choosing x at random in D and continuing with the protocol, B chooses an element $y \in D$, setting $X = f^{l-1}(y) = f^{l+1}(x)$, where now x is an element which he does not know. However, B *does* know all but the first bit of $\text{pad}(x)$. A, suspecting nothing amiss, sends B the encryption $\text{pad}(x) \oplus M$, all but the first bit of which B can easily decode. If the context of the message allows B to infer the value of that first bit, then he has learned the value of the hard bit $b_1 = B \circ f^{-1}(y)$ of a number y of his choice. Since our original assumption was that inverting f is efficiently reducible to $B \circ f^{-1}$, this is a successful attack.

$$\begin{array}{ccccccc}
 x = f^{-2}(y) & \rightarrow & f^{-1}(y) & \rightarrow & y & \rightarrow & \dots & \rightarrow & f^{l-2}(y) & \rightarrow & X = f^{l-1}(y) = f^{l+1}(x) \\
 & & \downarrow & & \downarrow & & & & \downarrow & & \\
 \text{pad}(x) = & & b_1 & & b_2 & & \dots & & b_l & &
 \end{array}$$

With the implementation using the bit-generator of Goldwasser, Micali, and Tong (see section 2), we now show how to refine Protocol 2 so that the resulting scheme is chosen-ciphertext secure. This refinement, based on the work of [11], is due to Silvio Micali.

Protocol 4

- $A \rightarrow B$: "Hi"
- B chooses at random x, x_1, x_2, \dots, x_n in \mathbb{Z}_N^* ,
and computes $\text{pad}(x)$, $X = f^{l+1}(x)$, and $X_i = f^{l+1}(x_i)$ ($i=1, \dots, n$)
- $B \rightarrow A$: X, X_i ($i=1, \dots, n$)
- $A \rightarrow B$: a random subset $S \subset \{1, \dots, n\}$ of size $n/2$

- $B \rightarrow A: \{x_i | i \in S\} \text{ and } \{xx_j \bmod N | j \notin S\}$
- A checks that $f^{t+1}(x_i) = X_i$ for $i \in S$ and $f^{t+1}(xx_j) \equiv XX_j \bmod N$ for $j \notin S$;
if so then $A \rightarrow B: M \oplus \text{pad}(x)$,
otherwise A halts (detecting cheating)

This protocol is secure against an eavesdropper; furthermore, it is secure against chosen-ciphertext attack by B. The protocol ensures that if A does not halt the transaction, detecting cheating, then with very high probability B has not cheated. In fact, the refinement may be regarded as a protocol during which B proves to A that he knows the number x , without gaining any additional knowledge -- for example, about the integer N . (Such a protocol is called a *minimum-knowledge interactive proof system* [11, 7].)

The same protocol will work, *mutatis mutandis*, as long as D is a group (in which we can compute efficiently) and f is an automorphism of the group.

We now sketch another solution to the problem of the chosen-ciphertext attack. In order to avoid any chance that A's public key be compromised by B's clever choice of random input to a probabilistic encryption protocol, we require that A choose the input; we proceed as follows. The solution has two stages. In the first stage, A chooses a sequence of random bits and transmits them one by one to B; these transmissions, of course, must be cryptographically secure. This can be accomplished by means of the minimum-knowledge protocol introduced in [7]; following this protocol, A can prove to B the value of a Boolean predicate in such a way that no eavesdropper can tell whether that value is 0 or 1, and so that B gains no additional knowledge at all. In the second stage, the sequence of bits can be used as the seed for a pseudo-random bit-generator; in this way, A and B simulate a shared one-time pad (of length polynomial in the length of the seed exchanged in the first stage).

6. Conclusions

By extending the capabilities of public-key encryption, we have demonstrated the power that interaction adds to the capabilities of cryptographic systems. Further study of interactive protocols promises to be of great use to cryptography. We take note here of the work of Rackoff in rigorous modeling of cryptosystems, including interaction [14].

The problem remains open of specifying a cryptosystem which does *not* use interaction and proving it secure against chosen-ciphertext attack. Perhaps a first step in this direction would be to allow some sort of limited interaction. For example, in the signature scheme of [10], which is secure against the analogous attack, the dependence of any signature on the history of previously signed messages (or on a random-function construction) may be regarded as an instance of interaction with that history (or with the random-function generator).

Acknowledgments

We would like to thank Silvio Micali for his encouragement of and major contributions to this work.

References

- [1] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr.
RSA/Rabin bits are $1/2 + 1/\text{poly}(\log N)$ secure.
In *Proc. 25th FOCS*, pages 449-457. IEEE, 1984.
- [2] Angluin, Dana and Lichtenstein, David.
Provable Security of Cryptosystems: a Survey.
Technical Report YALEU/DCS/TR-288, Yale University, October, 1983.
- [3] L. Blum, M. Blum, and M. Shub.
A simple secure pseudo-random number generator.
In *Crypto '82*. 1982.
- [4] Blum, M. and Micali, S.
How to generate cryptographically strong sequences of pseudo-random bits.
In *Proc. 23rd FOCS*, pages 112-117. IEEE, 1982.
- [5] M. Blum and S. Goldwasser.
An efficient probabilistic public-key encryption scheme which hides all partial information.
In *Crypto '84*. 1984.
- [6] W. Diffie and M.E. Hellman.
New directions in cryptography.
IEEE Trans. on Inform. Theory IT-22:644-654, November, 1976.
- [7] Z. Galil, S. Haber, and M. Yung.
A private interactive test of a Boolean predicate and minimum-knowledge public-key cryptosystems.
In *Proc. 26th FOCS*. IEEE, 1985.
- [8] S. Goldwasser and S. Micali.
Probabilistic encryption and how to play mental poker keeping secret all partial information.
In *Proc. 14th STOC*, pages 365-377. ACM, 1982.
- [9] S. Goldwasser, S. Micali, and P. Tong.
Why and how to establish a private code on a public network.
In *Proc. 23rd FOCS*, pages 134-144. IEEE, 1982.
- [10] S. Goldwasser, S. Micali, and R.L. Rivest.
A "paradoxical" solution to the signature problem.
In *Proc. 25th FOCS*, pages 441-448. IEEE, 1984.
- [11] S. Goldwasser, S. Micali, and C. Rackoff.
The knowledge complexity of interactive proof systems.
In *Proc. 17th STOC*, pages 291-304. ACM, 1985.
- [12] R.M. Needham and M.D. Schroeder.
Using encryption for authentication in large networks of computers.
Communications of the ACM 21(12):993-99, December, 1978.
- [13] M. Rabin.
Digitalized signatures and public-key functions as intractable as factorization.
Technical Report LCS/TR-212, MIT, January, 1979.
- [14] C. Rackoff.
Cryptography: lecture notes.
1985.
- [15] R.L. Rivest, A. Shamir, and L. Adleman.
A method for obtaining digital signatures and public key cryptosystems.
Communications of the ACM 21(2):120-126, February, 1978.

- [16] G.J. Simmons.
Symmetric and asymmetric encryption.
Computing Surveys 11:305-330, December, 1979.
- [17] A.C. Yao.
Theory and applications of trapdoor functions.
In *Proc. 23rd FOCS*, pages 80-91. IEEE, 1982.
- [18] M. Yung.
Cryptoprotocols: subscription to a public-key, secret blocking and the multi-player mental poker game.
In *Crypto '84*. 1984.

SOFTWARE PROTECTION:

MYTH OR REALITY?*

James R. Gosler
Sandia National Laboratory
Division 7233
Albuquerque, N.M. 87185

Abstract:

Staggering amounts of commercial software are marketed to fulfill needs from the PC explosion. Unfortunately, such software is trivial to duplicate! From the vendors' viewpoint a way to protect profit is needed. Typically, they have resorted to various schemes that attempt to inhibit the duplication process.

Although protection of future profit is important, so is protection against current loss. Commercial and business related software must be adequately protected lest data be stolen or manipulated. However, more important than any of these classes is protection of government computer resources, especially classified and operational software and data. Loss of control in this realm could be detrimental to national security.

This paper addresses current technologies employed in protection schemes: signatures (magnetic and physical) on floppy disks,

*This work performed at Sandia National Laboratories, supported by the Department of Energy under contract No. DE-AC04-76DP00789.

Software Analysis Denial (SAD), Hardware Security Devices (HSD), and Technology Denial Concepts (TDC) are presented, with an emphasis on SAD. Advantages and disadvantages of these schemes will be clarified.

1.0 INTRODUCTION

Software piracy, unauthorized penetration and system modification[12,13] are areas of threat to government and business computer systems, even the economic survival[6] of many software vendors is in peril. Vendors are typically using three main strategies to combat the piracy dilemma. The strategies[16], usually used in combination, include marketing, legal, and technological. A typical marketing strategy is to price software at an extremely attractive figure in the hopes that each potential customer will purchase it, especially to receive the required documentation and any technical consultation. The legal ploy[7] includes suing for copyright or licensing agreement violations. These schemes by themselves have limited effect, but are useful in combination with other strategies. Technological schemes are extremely varied in detail, though they can be typically grouped into a few categories. The effectiveness of these technical schemes vary substantially and this is a major topic of the paper. The technological arena provides the only substantial methodology to combat software threats in government and business application fields.

Concerns other than just preventing duplication of software are very important. For instance, software vendors may wish to protect against disclosure of proprietary algorithms, banking executives must prevent their system programmers from being able to examine or modify bank accounts, and government entities must design defense systems software with an intrinsic ability to prevent tampering of critical components or information. There are many examples where the hiding of critical information in software or detection of modified software is desirable, possibly mandatory. These conditions are found in all

types of computer related applications, but have yet to receive the attention they deserve. One of the greatest flaws current banking and government computer security systems have in common is an implicit assumption that the adversary does not have access to a system. There are many recent examples that show this is not a good assumption! Obviously, this assumption, concerning potential adversary access, is not the case with copy protection schemes where the system is essentially thrown into a den of "wolves".

The overall intent of this paper is to discuss current capabilities of both the defense and the threat, provide a comparison between them, and suggest a set of goals for the ultimate software protection system.

Examples used in this paper are fictitious, but at the same time are representative of current copy protection techniques. However, we will neither deal with the issue of how protection schemes impact the end user nor address any specific defeat methodologies to compromise current security schemes. The IBM PC, used by the author, will serve as a vehicle for examples.

2.0 FUNDAMENTAL CONCEPTS

We will provide some fundamental principles of copy protection and relate these via an analogy before discussing some of the technical concepts associated with defense and threat.

There are two broad components in a copy protection scheme. The first is a uniqueness associated with the system, which must be difficult to reproduce. Typically this is done with an unusual sector(s) on a floppy disk or by using a Hardware Security Device (HSD) that is separately attached to the system. The other component is special software that is usually embedded somewhere within application software and it is responsible for interrogating the presence of the uniqueness in the system. If present the special software may also determine if the uniqueness is pristine or altered.

A security or copy protection system of this type can be beaten by two general methods; by duplicating the unique signature associated with the system or by modifying the software (application or system) in such a manner that application software will operate without the unique signature being present. The adversary need use only one of the above methods and generally the easiest is chosen.

Thus, not only must the defense design a difficult to duplicate unique signature but, must also make it hard for an adversary to analyze and/or modify the software. Interestingly, many software packages, which employ some form of copy protection, do nothing to make analysis and/or modification of software difficult for the adversary -- these schemes are easily defeated. Importantly, it is this component of a good security scheme that has many applications outside the field of copy protection.

Another way of looking at the analysis and modification problem is by analogy using a burglar alarm. Suppose that a valuable asset must be protected and to do this we place the asset in a vault and surround it with an alarm system having many different types of sensors. These sensors are responsible for detecting changes to the normal operating environment. Upon detection, the alarm will be triggered, guards will appear, and the burglar will be permanently detained. In this analogy the valuable asset could be special software that checks for the presence of the unique signature. The sensors could also be special software which attempts to determine if the operating environment has been modified due to the use of, for instance, dynamic analysis tools by the adversary. Finally, the alarm might be anything from displaying an **UNAUTHORIZED DUPLICATE** message to implanting a **WORM** (software which will cause harm to the system) in the software.

For the software case it is feasible that an adversary has analysis tools with special properties that will not alter the monitored environmental parameters. Thus, the software can be analyzed easily without detection. In most cases even if detection occurs nothing terrible happens. Guards do not appear nor are worms implanted. The adversary, therefore, has an unlimited try capability and through repeated experiments will eventually win.

3.0 THE DEFENSE

A vendor wishing to protect his system will want duplication of the unique signature to be difficult for an adversary. He will also want it to be difficult to analyze or modify the software, which could bypass the need to duplicate the unique signature. Even the necessity of hiding proprietary algorithms may be appropriate. Government system software designers have similar objectives, but for different reasons. In order to prevent the adversary from having a working model of the system with which he can perform analysis at his leisure, designers want duplication to be difficult. However, the most important objectives are to make it extremely arduous to modify the system in a way that bypasses critical features (checks) and to obtain sensitive information (e.g. crypto variables). For both cases the objectives of the defense are threat scenario dependent. As such, designers must consider the ways in which their systems are vulnerable to an adversary and then take steps to thwart or nullify adversarial intrusion.

It is apparent that software developers with diverse applications have similar needs from the realm of software protection although the reasons they need protection are as diverse as the applications.

How can the defense achieve his objectives? As a vehicle to unveil techniques and concepts, we will employ an IBM PC as the system and copy prevention as the objective.

3.1 UNIQUE SIGNATURE

To make duplication of software difficult there must be an additional component(s) included in the system specifically to provide trouble for an adversary to reproduce. This component(s) usually falls into one of two categories in the commercial world.

The first category is comprised of a unique signature on the floppy disk itself. Typically, this comes in both a physical and a

magnetic form. The physical signature involves removing a small amount of magnetic material from the floppy disk surface with a laser. The scheme is implemented by software that writes some information to this damaged area and then reads the information from the same area back into memory. If the information read is the same as that which was written, then clearly there was no laser damage on the disk at the proper location. We can conclude the software/disk combination is not the original.

A magnetic form of floppy signature involves altering the standard IBM System 34 (double density)[11,18] recording format. Besides end user data, each track contains address marks, gap bytes (sync fields), sector ID fields, Cyclic Redundancy Check (CRC) bytes, and clock bytes. All of this information must be present and correct in order for the Intel 8272A floppy disk controller (FDC)[9,14] to properly process the end user data. It is quite possible, by altering this standard format[5], to cause the FDC to return an error status message back to the microprocessor (Intel 8088)[9,14,15] as a result of a disk operation. Examples of typical errors are bad CRC, sector not found, and address mark not found. For the system to determine that the unique signature is present, the software need only perform a disk operation(s) and then determine if correct error status is returned. It is also possible to create a non-standard disk format by issuing an unusual sequence of commands to the FDC or by using special hardware which bypasses the FDC and its inherent limitations.

The second category of unique signature consists of a hardware security device (HSD)[17], which is currently being used in many of the more expensive software packages. The HSD can be connected externally to the PC via the RS 232 port, the parallel port, or even placed in series with the keyboard. It is rarely connected internally since it typically requires use of a valuable card slot.

The manner of HSD implementation within a system also varies. In its simplest form the system will send a fixed value to the HSD which then responds with a fixed value. The software will compare the response with a stored value to determine if the HSD is present. In more sophisticated versions, the system will send to the HSD and receive from it a variable value, and possibly even have part of the

software encrypted and stored in the HSD. HSD advantages (from a security point of view) are that it is more difficult to duplicate than a floppy signature and it is not as obvious to an adversary when the system is looking for the unique signature.

The HSD is much more expensive, which constitutes its primary disadvantage, and therefore is usually not used in cheaper software.

3.2 SOFTWARE ANALYSIS DENIAL (SAD)

What is currently being done in commercial software to make adversarial analysis and/or modification of the software more difficult?

In the copy protection game, it does little good to have a unique signature, impossible to duplicate, if the adversary can easily modify software such that the signature need not be present for proper operation. Consequently, it is imperative to have a well balanced protection scheme -- the difficulty of duplicating the signature should be comparable to analyzing and modifying the software.

Since the most common tool used by an adversary is a software debugger, we will limit remarks to techniques the defense can employ to make analysis from this source more difficult. However, we will also address some techniques being used to make modification of critical or non-commercial software difficult.

Given that the defense knows what tool(s) the threat will likely use, he must determine how the normal operating environment will be altered by use of these tools. For the case of a debugger, there are available several modifications to the environment.

The first and most obvious change is that the debugger must reside in the same memory space as the application, thus, a foreign presence can be checked for. Typical debuggers depend heavily on certain interrupt vectors to single step and breakpoint the application software. Application software then could easily integrate the

use of these vectors into the application itself and thereby create difficulty in using the adversary debugger. Quite often in the analysis effort, it is convenient for the adversary to modify registers and/or memory locations to help in understanding of the software. Difficulty can be enhanced by making the proper execution of software highly sensitive to not only memory location and registers used, but to all memory locations and registers available. Finally, the application should have code which is timing sensitive because analysis of the software will alter its correct timing.

Assuming that the adversary can, through analysis, determine what he needs to modify, then the defense needs to employ techniques to make the desired modification difficult. The most common technique seems to be through use of checksums. If the defense realizes where an adversary will likely modify the software then they will perform checksums on this area of code hoping that any change to the critical code will alter the value of the checksum. Encrypting the critical software is another technique. If the adversary, through analysis, examines the decrypted form of the critical code and determines what needs modification, then he also must determine how to alter the cipher text that will yield the desired result. Public key cryptography is useful in this area. For example, if the algorithm used to encrypt the critical software was RSA and only the decrypt key was stored in the system, then the adversary would have an extremely difficult time determining how to change cipher text to achieve the desired plain text.

Numerous other techniques are currently being used in the commercial world, such as executable software movement, searching for breakpoint instructions and taking advantage of the Intel 8088 pre-fetch queue.

3.3 TECHNOLOGY DENIAL CONCEPTS (TDC)

Assuming acquisition of a working system, it is imperative to keep the adversary from performing dynamic analysis on it in an interactive fashion. If he is allowed to perform this interactive

dynamic analysis, he will eventually be able to locate and bypass all of the SAD features discussed in 3.2.

For this reason the adversary must be made to pay a penalty each time he is detected by SAD sensors. This penalty could be anything from destroying critical system components that would disallow further testing with that particular system to subtly altering the system in such a way as to provide disinformation, which is of no pertinent value, to the adversary.

Unfortunately, from the pure security viewpoint any commercial product containing or even suspected of containing TDC will suffer exceedingly due to consumer abhorrence and consequent economic leverage. This was typified by the irate consumer response directed against several software security vendors who boldly announced the intended use of worms in a future release of their products.

4.0 THE THREAT

Adversarial objectives of the threat are diverse. They encompass pirating commercial programs, subverting banking or government software, and stealing software-based proprietary algorithms. For example, suppose that companies A and B both produce and market an RSA encryption program. Further suppose that B's product is substantially slower than A's version primarily due to the speed of the algorithm responsible for finding large prime numbers. Company B's programming analyst could acquire a copy of A's program, reverse engineer the software, and then "borrow" the faster algorithm.

For government and perhaps business applications, the adversary's objectives are similar. Suppose the military has a computer based weapon control system. Part of its system software, responsible for access control, is password protected. To access the control system that will allow use of the weapon system without knowledge of a legitimate password, or to deny use of the weapon system to an authorized user the adversary must acquire tools to

duplicate the software and then determine what modifications would be necessary to alter the control system.

Before we discuss tools with which the threat attempts to accomplish his objectives, we need to provide a working definition of the adversary. The threat can be subdivided, in general, into insider and outsider categories with authorized access being the main difference between the two. The insider threat could be anyone from the designer of the system to an authorized end user of the fielded system. Thus, the insider threat can be broken into two categories: those intimately knowledgeable with the system, such as the design team, and those with little or no knowledge but having authorized access. However, this paper will not address the problem associated with the threat being part of the design team.

Since access control is typically a separate security issue, the outsider threat scenario considered will usually be under the assumption that the threat has already gained access to the system. Thus, for the purposes of this paper, a conservative approach is taken in that both the insider and outsider threat are considered essentially equivalent.

4.1 THREAT TOOLS

Adversarial tools that threaten commercial and perhaps other software fall into two main categories: tools used to duplicate the unique signature or its effect, and tools used to analyze software and hardware. They vary from no cost to \$100K+ and are readily available.

If the unique signature is an unusual magnetic encoding on a floppy, then there are many commercial products available that will analyze the floppy and attempt to replicate the signature. However, these software tools share one deficiency: all utilize the FDC for their analysis and duplication efforts. But, there exists unique signatures that are currently being used that were not created using the FDC and all its limitations. For example, some vendors use special hardware that will generate "weak bits". These bits are

impossible to duplicate using the FDC and are thus felt to be more secure by the vendors. Unfortunately, there are also available products[3] capable of separately encoding each bit cell on a track and at a variable flux density. Such products make duplication of all magnetically encoded unique signatures on floppy disks effectively trivial.

Even if the signature is a result of physical damage to the disk, the adversary has several options. First, with appropriate equipment, he can attempt to duplicate the physical damage, which could be difficult even with expensive equipment. However, depending upon the motivation and resources of an adversary, it is certainly feasible. A simpler and cheaper approach would be to alter the system so that software which checks for damage is "fooled" into thinking that the damage is present. This might be done by front-ending certain interrupt vectors which are tied to the FDC. Such front-end software would change the status of the FDC command to the correct and expected values.

If an HSD is installed as the signature then attacks similar to the physical damage case could be employed. Usually it is a straightforward task to alter software that is communicating with the HSD using a technique that renders the HSD needless. A much more complicated technique for defeat would be to duplicate the HSD. However, this addresses the tools and techniques of analyzing and duplicating microcircuitry, which is beyond our scope.

The adversary will make use of two general classes of tools in his analysis effort: static and dynamic. Assuming he has acquired use of the system, the adversary will use these tools against the binary form of software. For example, static tools can be used to locate all branching instructions and/or all occurrences of an INT 13H (disk operation) instruction. These classes of tools can often provide a good starting point for application of dynamic analysis tools. Actually, the more structured the programming methodologies the more straightforward it is to use these tools. This situation is certainly better for the adversary.

Dynamic analysis tools are the real workhorses for the adversary. They include software debuggers[1,2,4], in-circuit

emulators (ICE)[8,10], and simulators. We have determined that the software debugger and ICE type tools are particularly useful for analyzing software systems.

These dynamic tools allow the adversary to execute the software in a controlled fashion. That is, the software can be executed one instruction at a time (single step). Then between instructions the analyst can examine/modify registers and/or memory locations. In addition to the single step mode, the analyst can also stop processing as a function of several other types of events. For example, execution could be halted and the environment examined/modified when:

1. Instructions are fetched or executed
2. Operands are fetched or modified
3. I/O ports are referenced
4. Memory/register contents reach predetermined values

Simple but powerful tools such as these give the adversary an enormous amount of information and consequently, it becomes a nearly straightforward task for the analyst to wade through the software to achieve his objective. The only difficulty that the analyst must be aware of is modification of the operating environment in a way that will trip a security sensor. Fortunately, for the adversary, even if he trips a sensor there will not be a debilitating penalty in most current systems. Thus, through an iterative process he will eventually work his way through or around all the sensors on the path to his objective. If the adversary's tools modified the environment in a detectable fashion and a significant penalty were imposed then the adversary is forced to proceed at a far slower pace. He must execute smaller blocks of code before hitting a breakpoint and he must also attempt to fix any environment modifications. Depending on the payoff, however, the adversary may well be willing to pay this extra price to analyze systems using penalties.

5.0 THREAT VS. DEFENSE

We have briefly discussed the objectives, tools and techniques of the two players. Our purpose here is to point out some strengths and weaknesses of the schemes currently being used commercially.

Many advantages in this game reside with the threat. As always, the adversary plays his cards last and thereby gets to attack a static security design. Beyond this there is another major obstacle for any cryptographic solution to the security dilemma. Even though the defense uses cryptographic schemes to scramble the executable software he must include not only the decryption algorithm but also all of the necessary cryptographic keys as part of the system.

The weakest characteristic of these schemes is the fact that the adversary never has to pay a penalty and in effect has an unlimited number of tries in order to achieve his objectives. To make matters worse some schemes typically broadcast to the outside world that a security violation has occurred. They will usually provide for the adversary a detailed road map to the sensor location. This weak characteristic alone makes defeat of these security schemes significantly easier!

Currently, many systems use a security front-end to their application software. This is done for several reasons, one of which is that it does not require modification to the application software, which makes the addition of protection easier for the vendor. Unfortunately, these front-ends are typically very easy to completely remove leaving the adversary with the unprotected application. Also, due to the proliferation of security schemes numerous software vendors purchase and use the same security package. Consequently when one package is defeated the rest will fall in short order and with minimal effort.

As previously stated, it appears that all of the more advanced security schemes rely on the use of clever programming tricks to detect an adversarial presence. This tends to make the reverse engineering process more difficult. It is not clear, however, as was pointed out in Simmons[19], how effective these defensive tricks can be designed to preclude or significantly delay the adversary from ultimately achieving his objectives.

Fortunately, there are several techniques that could be employed to make software analysis/modification more difficult. Most importantly, the adversary must be made to pay for his mistakes. A suitable penalty in the commercial world would simply be to make the

application software nonfunctional. The best way to alter the software to a nonfunctional state is to cause the software to fail intermittently with subtle problems. For example, suppose the XYZ corporation produces and markets a CAD/CAM program. Upon detection of an adversarial presence the penalty to be invoked might be to alter the software so that the drawings sent to a plotter will randomly miss pen strokes. In the case of spreadsheet software, the numerical calculation associated with the spreadsheet columns could be subjected to random errors.

With proper implementation of this type of penalty the adversary will not be tipped off that he has been caught. Later when he or his customer is using the application software there is a good chance that he will not associate the sporadic (flaky) operation to the pirated copy. This sort of tactic prevents another and perhaps more intensive attack on the target software.

Many other techniques could be used to improve the security of software using current methodologies. However, we feel they, at best, provide very limited protection from a sophisticated opponent. The security of the system should not depend heavily on how cleverly the designer implemented his tricks. An enormous need exists for software security systems that provide a high degree of predictable protection. What we really need are methodologies whose security is comparable to that of a good cryptographic system.

6.0 RESEARCH GOALS

Research applications in this area will impact software based systems in four distinct domains: 1) security level 2) cost 3) reliability 4) performance. Obviously, the optimum objective of SAD/TDC is to provide maximum security at minimal cost, with no impact on system reliability or degradation of system performance. This is an impossible task. However, depending on the application, the above optimum objective could be relaxed and realistic requirements could still be achieved.

Ideally, the level of security provided by SAD/TDC is equivalent to security associated with modern cryptographic based systems. That is, the compromise of a system should be dependent on the adversary dealing with the computational complexity issue. As mentioned, however, all SAD/TDC currently being employed involve the use of clever tricks and attempts to conceal information from the adversary. After refinement, perhaps even these techniques may be adequate for limited situations. For example, suppose our secure system, which we have control over, is one in which the unique information (a special algorithm perhaps) can be made obsolete within one week after detecting loss. If so, a security system which provides at least two weeks of delay to the adversary may be adequate. In the limit then our secure system could have its uniqueness changed inside the cycle time of an adversary.

Costs of these sorts of systems can be broken into three categories: 1) development 2) production and 3) administration. Development, that is the cost to design and integrate the security subsystems into the applications, is a one-time item and thus, this cost increase will usually have the most latitude. However, additional production costs will be incurred for each system produced and as such, may receive much closer scrutiny from management. On the other hand, in an extremely high security application, as is the case with control for nuclear weapons, costs become of secondary importance, so an increase of perhaps a few thousand dollars for the security system becomes acceptable. Administrative costs are associated with maintaining system security requirements, such as the need for key management. Costs such as these are recurring and potentially substantial. The security designer should always be attentive to this area.

Many of the application systems that need added security have requirements for extremely high reliability. For this reason the security designer must be very careful with use of certain techniques such as timing tricks. Many times a security system undergoes an independent review process, which is designed to determine if subversive features (trapdoors, trojan horses, etc.) are present. Unfortunately, it may be more difficult to detect designer induced subversive constructs due to current techniques being used by the designers to improve system security.

Based on this current state of affairs, if the software designer were a covert agent, then he could compromise the integrity of the system while appearing to increase its security!

Finally, the issue of system performance can be of overriding importance depending on the application. The key idea is to minimize or eliminate such adverse effects. If an application is the guidance subsystem software for a defensive missile then performance degradation could not be tolerated. For instance, the systems reduced capability to update the missiles parameters may result in an unacceptable reduction of kill ratio.

Now, we would like all parameters of a security subsystem skewed in our favor in an optimal fashion, but realistically this does not seem feasible. Compromises will need to be made with the security design as a function of application system requirements so that an optimum balance is achieved.

7.0 SUMMARY

Considerable effort and resources are expended to prevent "hackers" or outsiders from attaining illegal access to computer systems. The same is not true, unfortunately, concerning the insider adversary having access to a computer system. Partial or complete access can lead to unauthorized duplication or modification of the systems software.

Current defense methodologies are not adequate to prevent or even significantly delay an insider adversary from achieving unethical to illegal objectives. Many software applications in both business and government sectors are in dire need of effective techniques to thwart an insider or outsider (who has acquired access) attack. Although the level of security offered through current methodologies can be enhanced to some degree, the results will still be unsatisfactory because the problem stems from these marginal methodologies.

We must provide new developments that, for example, explore the world of cryptology and exploit the limits of numerical complexity to the extent the security of a system is provable, or at least predictable. With this sort of focus perhaps, the myths of software protection and security can be transformed to reality.

8.0 REFERENCES

1. S. Armbrust and T. Forgeron, "Entymological Explorations", PC Tech Journal, vol. 3, no. 1, (Jan. 1985), pp. 88-109.
2. S. Armbrust and T. Forgeron, "Untangling Problems", PC Tech Journal, vol. 3, no. 4, (Apr. 1985), pp. 81-95.
3. COPYIIPC Option Board Manual, Central Point Software, (1985).
4. D. Daftwyler, "Professional Debugging", PC Tech Journal, vol. 3, no. 3, (Mar. 1985), pp. 60-73.
5. Disk Mechanic Technical Manual for the IBM Personal Computer, MLI MICROSYSTEMS, (May 1985).
6. D. Gabel, "Copy Protection", PC Week, vol. 2, no. 34, (Aug. 1985), pp. 35-37.
7. G. Geruaise Davis III, Esq., Software Protection: Practical and Legal Steps to Protect and Market Computer Programs, Van Nostrand Reinhold, New York, (1985).
8. HP 64000 Logic Development System Model 64620A Logic State/Software Analyzer Reference Manual, P/N 64620-90903, Colorado Springs, HEWLETT-PACKARD, (1982).
9. IBM Personal Computer Technical Reference Manual, IBM, (Apr. 1983).
10. I²ICE Integrated Instrumentation and In-Circuit Emulation System Reference Manual, P/N 163252-003, INTEL, (1984).
11. ISBC 204 Flexible Diskette Controller Hardware Reference Manual, P/N 9800563-02, INTEL, (1979).
12. B. Landreth and H. Rheingold, Out of the Inner Circle: A Hacker's Guide to Computer Security, Microsoft Press, Bellevue, Washington, (1985).
13. S. Levy, Hackers Heroes of the Computer Revolution, Anchor Press/Doubleday, Garden City, New York, (1984).
14. Microsystem Components Handbook: Microprocessors and Peripherals, vol. 1&2, P/N 230843-002, INTEL, (1985).

15. S. Morse, The 8086/8088 Primer: An Introduction to their Architecture, System Design, and Programming, Hayden, Rochelle Park, New Jersey, (1982).
16. D. Parker, Fighting Computer Crime, Charles Scribner's Sons, New York, (1983).
17. W. Rosch, "Internal Security", PC Week, vol. 2, no. 18, (May 1985), pp. 89-108.
18. Shugart OEM Manual: SA 810/860 Single/Double-Sides Half-Height Diskette Storage Drives, (1982).
19. G. Simmons, "How to (Selectively) Broadcast a Secret", Proceedings of the Symposium on Security and Privacy, Oakland, California, (Apr. 22-24, 1985), pp. 108-113.

Public Protection of Software

Amir Herzberg and Shlomit S. Pinter

Dept. of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, ISRAEL

Abstract

One of the overwhelming problems that software producers must contend with, is the unauthorized use and distribution of their products. Copyright laws concerning software are rarely enforced, thereby causing major losses to the software companies. Technical means of protecting software from illegal duplication are required, but the available means are imperfect. We present protocols that enables software protection, without causing overhead in distribution and maintenance. The protocols may be implemented by a conventional cryptosystem, such as the DES, or by a public key cryptosystem, such as the RSA. Both implementations are proved to satisfy required security criterions.

1. Introduction

Great losses to software producers are currently incurred due to the ease of copying most computer programs. It is common practice for one user to buy a software product, and without the producer's consent to give or sell it to other installations. The economic value of software protection resulted in many products that supplied means to protect software. It is shown in [HK84] that many commercially available means suffer from some of the following deficiencies:

1. Insufficient protection.
2. Impaired backup capability (for the innocent user).
3. Narrow range of applicable systems (i.e., methods that protect only firmware).
4. Obstacles for distribution and maintenance, of the computers and the software.
5. Excessive overhead in total costs or in execution time.

This paper describes and proves the security of a software protection system that does not suffer from the deficiencies indicated. A preliminary version of PPS (Public Protection of Software) has been presented, with other software protection methods, in [HK84]. In contrast with the deficiencies outlined above, PPS provides:

1. Provable, hence reliable, protection.
2. Undisturbed backup capability.
3. Applicable on virtually all systems.
4. Simple, undisturbing protocols for distribution and maintenance.
5. Reasonable overhead in total costs and execution speed.

PPS requires modifications to the architecture of the processor. Therefore, it can only be implemented by CPU manufacturers. In a recent paper [AM84], another software protection method (henceforth referred to as AM) was presented that requires similar modifications to the internals of the processor. PPS differs

mainly in the protocols used. The PPS protocols require less communication between the parties, and minimal intervention of the key generating body (denoted Z) and the software producer. For example, communication between the software producer and the system integrator before the protection of each product is not required. This communication is essential in AM. In addition, PPS provides protocols for replacing malfunctioning CPUs and indirect software distribution (via a dealer). AM does not provide protocols for those functions. A detailed comparison of PPS and AM may be found in Section 2.1.

PPS is the combination of three protocols, two for the distribution of software and one for replacement of malfunctioning CPUs. PPS may be implemented either by public key cryptosystems or by conventional cryptosystems. Section 2 discusses the protection supplied by PPS. In Section 3 we describe how PPS may be implemented by public key cryptosystems (PPS/PK). Section 4 a formal model for discussing the security of PPS is presented. The security of the public key cryptosystem implementation is then proved. This implementation is straightforward, but the conventional cryptosystem implementation (PPS/C) presented in Section 5 seem to be much more realistic. Section 6 gives the final conclusions. The protection provided by PPS

PPS attempts to render unprofitable the effort required to copy protected software. PPS relies upon mechanisms embedded in the CPU, therefore PPS cannot prevent the CPU producer from making secret trapdoors in the CPU that will enable software duplication. PPS requires a key-producing body, which installs the initial keys in the CPU and enables replacement of failing CPUs. This body may be the CPU producer, and it is represented by Z or the *center* in this paper. PPS enables Z to distribute the keys in such a manner that prevents other bodies from creating valid keys. If the system's OEM is Z , this feature might help to prevent the creation of "clones" (compatible computers by other OEMs).

Intuitively, PPS provides three levels of protection. The first level is against simple piracy attacks. Such attacks use legal procedures and attempt to duplicate

software by some unforeseen manipulation of those procedures. The second level is against more determined attacks, that include the faking of a CPU failure. Due to obvious reasons, a new CPU, that runs all the software bought for the failing CPU, should be provided quickly. It is obvious that if the CPU did not really fail, and is not returned, the attackers will have two CPUs that run the same software. While this hazard should be protected against by an appropriate procedure, PPS ensures that no further gain may be achieved by faking a CPU failure. PPS's third level of protection is against attackers that physically violate the CPU's enclosure, and discover (literally!) the keys held within. This approach is quite extreme, but it has been argued that such attacks may be attempted by parties that desire to cause distrust in the center or in the CPU. Only when implemented by a public key cryptosystem, PPS provides some protection against this attack. After violating the integrity of the CPU, the attackers will only be able to decypher protected code encrypted for the violated CPU.

A possible modification of PPS is transferring a key (*execution key*) instead of the actual program [AM84, HK84]. The program is then transferred and encyphered by that key. The CPU operates the program using the execution key. The security analysis of such a modification would not change compared to that of PPS (given in Section 4). As described in the references, this modification might improve greatly the performance of the CPU.

1.1. PPS vs AM

1. The influence of PPS on the architecture of the CPU is the same as the influence detailed in [AM84], and is not discussed here.
2. Both methods provide sufficient protection, undisturbed backup capability, wide range of applicable systems, and reasonable overhead in total costs and execution time.
3. PPS may be implemented either by using public-key cryptosystems or by using conventional cryptosystems, while AM requires public-key

cryptosystems. The implementation of public-key systems is much harder.

4. PPS does not require communication between the software producer and the customer during the purchase of the software. Rather, an untrusted dealer may sell the software, with no need for immediate communication with the software producer (see Section 3.2). This communication is essential in AM, and may present quite an obstacle in software distribution.
5. PPS does not require communication between the software producer and the system integrator before the protection of each product. This communication is essential in AM and presents another obstacle in software distribution. Also, the added transmissions may be tapped and altered, and the security is endangered.
6. PPS provides a protocol that enables the replacement of a malfunctioning CPU by untrusted servicemen, without requiring the physical transfer of a new CPU from the producer. AM requires the physical transfer of a new CPU.
7. The motives of all the parties involved in the usage of the protection method (CPU producers, system integrators, software producers, etc.) are similar in both methods. Those motives are discussed in depth in [AM84]. We will not repeat these arguments.
8. AM allows the system's OEM (Original Equipment Manufacturer) to require a fee from software producers for each usage of the system to protect software. By a simple variant to PPS the same result may be achieved. We will not discuss this here.

2. Implementation of PPS with Public-Key Cryptosystem (PPS/PK)

The implementation of PPS requires encrypting functions inside the CPU. The encryption may be done by a public-key cryptosystem (PKCS), such as [RSA78], or by ordinary encryption methods, such as [DES77]. In this section we will describe the implementation by a PKCS, denoted PPS/PK. This implementation is more straightforward; however, since no implementation of a PKCS seems both secure

and quick, the implementation by conventional cryptosystems seem to be more reasonable. The concept of PKCSs has been first suggested in [DH76], and several implementations - as well as numerous applications - have been published since then [M83].

A PKCS based on a set of pairs of functions $\{<E_i, D_i>\}$ such that

- C1. $D_i E_i = E_i D_i = 1$
- C2. Knowing $E(M)$ and E , but not D , does not reveal anything about M .
- C3. Knowing $D(M)$ and M does not reveal D .

We use E to denote the encrypting function (or key), and D or E^{-1} for the Decyphering function (or key).

With each computer unit u , associate a pair of keys $<E_u, D_u>$, and with Z associate a pair of keys, $<E_z, D_z>$. Every computer unit C_u contains the following information:

- 1. D_u - The decyphering (secret) key of C_u
- 2. E_z - The encrypting key of Z
- 3. $D_z E_u$ - The encrypting key of C_u , signed by Z .

For indirect distribution via a software dealer U , another key is required in the dealer's computer - $F_u(i)$:

- 4. $F_u(i)$ - The software producer sells his or her software to the dealer with this key. The key is changed between sales.

The keys D_u , $F_u(i)$, and E_z , are kept hidden inside the CPU itself. They may not be accessed by the CPU instructions, except the special instructions that implement PPS. The signature of Z , denoted D_z , is even more secret: it is not kept in the CPU at all. On the contrary, E_u may be used quite easily (and is not a secret).

The cryptographic utilities required for PPS are only trapdoor functions. Actually, we only require $D_u E_u = 1$ for every computer u , and $E_z D_z = 1$.

The cryptosystem may be commutative, i.e. $E_a E_b = E_b E_a$. Several PKCSs have this property, including [RSA78], and some protocols are not secure with commuting cryptosystems. If other properties are known for the cryptosystem, analysis as in Section 4 should be done.

2.1. Direct software distribution protocol (PPS/PK)

The protocol that a user U with computer C_u should follow in order to buy PPS/PK protected software from its producer P is the direct distribution protocol outlined below. Note that information should pass only once from the user to the producer and vice versa. The notation used for a user U sending a message M to his computer C_u or to another party B is: (U, M, C_u) or (U, M, B) respectively.

- D1. $(U, D_z E_u, P)$ - The user U sends to P the encryption key E_u signed by Z .
- D2. $(P, (D_z E_u, PGM), C_p)$ - The producer P enters the encryption key of the customer's computer signed by Z and the program to be distributed, PGM , into his computer.
- D3. $(C_p, [E_z D_z E_u] PGM, P)$ - The encryption procedure E_z is known to all computers, but hidden from the users.
- D4. $(P, E_u PGM, U)$ - The user receives the software package.
- D5. $(U, E_u PGM, C_u)$ - Loading the program.
- D6. $(C_u, O(D_u E_u PGM), U)$ - The computer C_u (but not U) knows D_u . While executing, the code PGM is hidden inside the processor. The operation (run) of software P by a computer is $O(P)$.

It is assumed that knowing $O(PGM)$ does not enlighten the intruder about PGM .

2.2. Indirect software distribution protocol (PPS/PK).

Usually software is not sold directly from the producer to the customer, but rather it is sold via a third party, the software dealer. Even telephone connection with the producer should, in these cases, be avoided. The direct software

distribution protocol, described in Section 3.2, is not suitable here, since the producer may rarely rely on the honesty of all the dealers. PPS provides a special protocol for indirect software distribution. This protocol requires one extra key hidden inside the dealers' CPU. The extra key is changing in each execution of the protocol. This temporal key, $F_u(i)$, (of a dealer U) is assumed to be the key of a conventional cryptosystem (although it could be implemented with a PKCS as well). The protocol is divided into two phases. In the first phase, the dealer L buys token programs from the producer. The tokens are converted to useful programs by the dealer's computer, C_L , in the second phase. Each token produces no more than one useful program, encyphered with the key of some buyer's computer. The initial key $F_l(0)$ is known only to the software producer. For example, $F_l(0)$ may be initiated in C_L by the producer before the computer C_L is given to the dealer.

The distribution protocol is outlined below. The first phase I1 is done for each token i to be used. Note that information should pass only once in each direction.

11. $(P, F_l(i)[PGM, F_l(i+1)], L)$ - The producer P gives the dealer L a token i . This is the first phase of the protocol, and it may be done independently of the other phases.
12. $(U, D_u E_u, L)$ - The user U sends his key to the dealer.
13. $(L, (F_l(i)[PGM, F_l(i+1)], D_u E_u), C_L)$ - The computer C_L now contains key D_u that corresponds to token i .
14. $(C_L, E_u PGM, L)$ - In the same time, C_L changes from key $F_l(i)$ to the new key $F_l(i+1)$. The new key that is given in the token!
15. $(L, E_u PGM, U)$ - From this step on, the protocol is the same as the direct distribution protocol. The user receives the software package.
16. $(U, E_u PGM, C_u)$ - Loading the program.
17. $(C_u, O(D_u E_u PGM), U)$ - The computer C_u (but not U) knows D_u . While executing, the code PGM is hidden inside the processor. Several $F_l(i)$ mechanisms may be implemented in the same processor. Also, processors dedicated to users need not have $F_l(i)$ at all.

2.3. The replacement protocol (PPS/PK).

If the CPU of a user malfunctions, a new CPU must be provided. An essential property of the new CPU is being completely compatible: every software run on the old CPU should also run on the new one. To enable the new CPU to run PPS/PK protected software, it must have the same keys as the old one. A similar requirement ensues from upgrades to the CPU, when CPU replacement is required.

The new CPU must be made available as soon as possible. It should be possible for several service centers to make available a CPU to replace any malfunctioning CPU in their territory. Obviously one cannot permit such service centers to produce CPUs and determine their keys at will. We present a solution in which deceptions are likely to be discovered or prevented, and even if deception is committed by the service center, no more than one illegal CPU will be obtained. Those results are formally proved in Section 4.3.

The solution we suggest to this problem requires the remote help of Z. However, this help is only remote (by communication), and does not require physical interaction with Z, as in [AM84]. The protection will not fail even if the communication is tapped or altered.

Every CPU replacement will require Z's intervention. After the CPU has been replaced, Z must verify that a replacement has in fact taken place (for example, by receiving the malfunctioning CPU and verify it's identity). The service center S uses the remote help of Z to convert a spare computer C_s (with keys E_s and D_s) into a replacement for C_u . After the successful completion of the protocol, C_s will have keys E_u and D_u . The replacement protocol is outlined below.

- R1. (U, D_s, E_u, S) - User U requires replacement CPU from S.
- R2. $(S, (D_s, E_u, D_s, E_s), Z)$ - The Serviceperson asks Z for a transformation key that will change the key of the spare CPU C_s from E_s, D_s to E_u, D_u .
- R3. $(Z, E_s(D_u; \text{replace}), S)$ - By the creation tables, Z finds for E_u the corresponding D_u . Then Z encrypts D_u - concatenated with a predefined string - by E_s , and sends it to S.

- R4. $(S, E_s(D_u, \text{replace}), C_s)$ - Installation of new key in C_s . The key D_u will be installed only if it is concatenated with the correct string. The public key $D_s E_u$ is installed too.
- R5. The CPUs may be replaced. The replaced CPU ought to be returned to Z and its number verified.

3. A Formal Analysis of PPS/PK

The presentation of any nontrivial security protocol or system would not be complete without a formal representation of the assumptions and formal proof of security. Therefore, we prove that, under acceptable assumptions, PPS/PK is secure. This is done using the Transaction System Model [HP85]. We proceed by describing the essence of the model and the correspondence between the model and PPS/PK. The model as described below is a simplified version of the transaction model for systems in which the timing is irrelevant to the security. Merritt [MB3] also presented a formal model for analyzing the security of protocols.

The formalization of cryptographic protocols enables a precise inspection of the arguments of security. In the case of PPS, the reader is encouraged to inspect if the formal model is truly derived from the assumptions and protocols, and if the proofs of the security of the model are valid.

3.1. The essence of the Transaction Model.

A *Transaction System* (TS) is a partial algebra, defined by a domain and a set of relations on that domain. The domain of a TS is considered as the set of all the possible states of some information system. A state is defined by a set of variables. One of the variables is the set of all the messages transmitted so far. The set of messages transmitted is known to the attackers, since they have complete control over the communication lines. A *state* S is a set of values of all the variables. The relations on the domain represent the possible inferences available for the attacker. The relations are grouped into meaningful sets, called *Transactions*.

Each transaction is a set of ordered pairs of states. A *Transaction System* $TS=(T,S)$ is defined by a set of transactions T on a set of states S .

The definition of a TS does not yet ensure that the TS represents the real world correctly. A TS would be *correct* if all the possible inferences for the attacker from a given state, and no impossible inferences, may be obtained by executions of transactions from that state. For example, inferences include the innocent activities of other participants, usage of properties of functions used, etc.

A pair of states (S_i, S_{i+1}) of a TS is an *ordered pair*, with S_i termed *Tail* and S_{i+1} termed *Head*, if S_{i+1} is the result of applying some transaction of TS on S_i . A sequence of states S_0, S_1, \dots is a *history*, starting from S_0 , if for all $i \geq 0, (S_i, S_{i+1})$ is an ordered pair. The *length* of a history is the number of states in the sequence. A state S_i is *i-reachable* from state S_0 if there exists a history H of length $i+1$ which starts at S_0 and ends at S_i , and no shorter history exists from S_0 to S_i . If there exists an i such that state S_i is i -reachable from state S_0 , then S_i is *reachable* from S_0 . If a state S_k is not reachable from state S_j , we say that S_j is *harmless* for S_k . A set of states is *reachable* if any of the states in the set is reachable. Similarly, we define the harmless property for a set of states.

We state without proof some elementary and intuitive results. The proofs are simple, and are given in [HP85].

Lemma 4.1 proves the transitivity of the reachability property.

LEMMA 3.1. *If a state S_i is i -reachable from S_0 , then every state S_j , j -reachable from S_i , is $(j+i)$ -reachable from S_0 .*

Theorem 4.1 proves that the results obtained will hold for more restricted cryptosystems, for example - without commutativity between cryptographic operators.

THEOREM 3.2. *Let S be a set of states harmless for a set of states D_i in TS , then in every TS' s.t. $Transactions(TS') \subseteq Transactions(TS)$, state S is harmless for D_i .*

3.2. PPS/PK as a TS.

The protocols detailed in Section 3 for PPS/PK execution correspond to the following TS called PPS/PK, under the assumptions listed below:

- A. Information hidden inside a processor cannot be read.
- B. Resurrecting the software by observing the ports outside the CPU is infeasible.
- C. The cryptosystems used are secure. The security requirements have been detailed in Section 3.
- D. The producer verifies faultlessly the identity of the user that sent the payment, and always delivers the software. The payment could have been implemented in the protocol, but it seemed unnecessary.
- E. No information leaks from Z (except by the replacement protocol).
- F. All the keys are cryptographically independent - no key may be obtained by known manipulations of other keys. The notion of cryptographic independence is formally defined in [HP85].

For proving the safety of PPS/PK we need consider only one producer of software, P . All the attackers may, however, use the protocol as if they are producers. For the analysis, assume that all the users are attackers (since the attackers can impose as honest users). The variables of PPS/PK are: X is the total expenses of the attackers, for every user u , K_u is the decyphering key of his computer C_u . Initially K_u contains D_u . During a CPU exchange, a K key of a spare computer is changed to the D key of the failing computer. For every dealer L , Q_L has the same rule as the K key for the temporal key $F_L(i)$. The set M of all the messages transmitted so far, which corresponds to the information held by the attackers.

The only source of information in PPS is the defined transactions. Therefore if PPS is in any given state, then that state is reachable from some initial state in which no messages were sent. The transactions of PPS/PK for computers C_u and C_w are listed in Table 1. In the table, P denotes a program to be sold by some

software producer for the sum of money - *cost* (by T12, T13 and T14). An application of operator a on string b is denoted $a(b)$. We omitted the brackets where there was no danger of confusion.

The TS model is a worst case analysis of the system. Therefore, data and keys are interchangeable (a key may be used as data and vice versa). Also, knowing the key of a cryptofunction is equivalent to knowing that cryptofunction. Therefore any string or key may be 'applied' to any string or key. This application may be done implicitly in some of the transactions, or directly by the attacker (by T7). When a transaction is explicitly used in one of the protocols, we note the step in the protocol. For example, T9 is used in D5 (step D5 of the distribution protocol).

The transactions basically represent the capabilities of the attackers. If an attacker manages to use some transaction with proper input, the table shows the output and the change in the system. The results of a transaction are added messages ("output") or a change for the variables X , Q or K . In the table, before any transaction is used, assume $K_u = D_u$ and $Q_u = F_u(i)$.

Some of the transactions will not be available in certain implementations. For example, the transactions that present the commutativity of the PKCS will not be present with a non-commuting PKCS. But, from Theorem 4.1 the security properties that were proved, hold as well without those transactions. Transaction T18, physically violating the CPU integrity, would not be considered part of PPS/PK. The TS that includes all the transactions, including T18, denoted as PPS/PKV, would be referred to only in the last theorem.

Notes: see the description of PPS and verify that all the steps in the protocols are performed by those transactions. We do not differentiate between operators and strings. When a string should be used as an operator, we use it as a key for the cryptographic operator.

A special kind of attack may be performed by an attacker which is also serviceperson. Such an attacker might accept replacement for a CPU from Z without returning the original CPU. This attack causes expenses to the attacker (including

risk); those expense are by denoted R . Theorem 4.6 shows that after using T17, there is no way to get more then two CPUs that use the same key (that originally belonged only to one of them). This ensures also that if the CPU have been replaced properly, the attackers will have only one CPU with the old key, and therefore with no gain.

Another extreme attack is physically violating the enclosure of the CPU, to find the keys hidden within (T18). The expense of this attack is denoted by V . Theorem 4.8 shows that when PPS is implemented by PKCS, even if T18 is used, the attacker must still use T12 with $E_z(E_u)$, where u is the identity of the attacker's computer, to obtain the uncyphered program P . This result enables enforcement of auditing means against such attacks.

3.3. Proofs of PPS/PK security.

The next lemma shows that no attacker can forge the signature of Z . The discussion in this section refers always to PPS/PK, except where stated otherwise.

LEMMA 3.3. *If $S=(M,X,K)$ is reachable from $S_0=(null,X_0,K_0)$, where $D_z a \in M$, then there exists computer C_u and b such that $a=E_u b$.*

Proof. Only T10 produces a message that includes D_z , therefore $D_z E_u$ must have been manipulated to produce $D_z a$. To remove E_u only T9 or T1 can be used. But the only result of T9 is operated by O and there is no transaction that removes O . The use of T1 to remove E_u requires an input string that includes D_u but not E_u . But no transaction produces such a string. Therefore $D_z a$ cannot be produced unless $a=E_u b$. ■

Theorem 4.4 shows that the attacker cannot reproduce the decyphered code P , given the encrypted program by T12 or T13. The producer's computer uses E_z on the input string x sent by the user, to produce the encryption for the program P . This is given by $[E_z x]P$ for any string x . Reproducing the encrypted program implies $P \in M$.

THEOREM 3.4 *If $S_1 = (M_1, X, K)$ is a harmless state for $W = W_a \cup W_b$, where $W_a = \{(M, X, K) \mid P \in M\}$ and $W_b = \{(M, X, K) \mid \exists D_u \text{ such that: } D_u \in M\}$, then $S_2 = (M_1 \cup [E_x x]P, X, K)$ is harmless for W .*

Proof. By contradiction, assume W is reachable from S_2 . Since S_1 is harmless for W , then $m_2 = [E_x x]P$ has been used to reach W . The only transaction, when D_u is unreachable, that removes E_x is T10, where $x = D_x E_u$. Therefore it remains to show that $S_3 = (M_1 \cup E_u P, X, K)$ is harmless for W . However, there is no transaction that removes E_u when D_u is unreachable. Thus, both W_a and W_b are unreachable, since both require the removal of E_u and D_x . ■

We have shown the original code cannot be obtained. Now we prove that the code cannot be 'adjusted' to another computer, i.e. no manipulation to the encrypted code produces code encrypted by a key of a different CPU. The idea of the theorem is that if an attacker can't get a program without paying, then he can't get two programs without paying twice the price of the program.

THEOREM 3.5. *If $S \in \{(M, Y, K) \mid Y < \text{cost}\}$ is a harmless state for some set of states U_1 defined below, then it is also harmless for U_2 . Where: $U_1 = \{(M, X, K) \mid (X < \text{cost}) \& (E_u P \in M) \& (K_i = D_u \neq \text{null})\}$ and $U_2 = \{(M, X, K) \mid (X < \min(2 * \text{cost}, R)) \& (E_u P, E_w P \in M) \& (j \neq i) (K_i = D_u \neq \text{null}) (K_j = D_w \neq \text{null})\}$*

Proof. If $E_u P \in M$, T12 or T13 must have been used. By Theorem 4.4, P is not in M . If T13 have been used to reach $E_u P \in M$ from S , then T14 must have been used before since it is the only transaction that produces $F_u(i)[P, F_u(i+1)]$. But if T14 occurred, it must have been in a history reachable from S , since $Y < \text{cost}$. In order to prove that U_2 is not reachable from S , we notice that T12 and T14 cannot be used twice. Also, from the arguments above, T13 cannot be used again. Therefore $E_w P$ cannot be produced by T12 or T13, and since no transaction that removes E_u it remains to show that no two computers can have the same key. That is for every two computers i, j where $i \neq j$, $K_i = K_j = D_1 \neq \text{null}$. In order to get a second key transactions T17 or T16 must be used. Since $X < R$ in U_2 , only T16 can be used, but the application of T16 change K_u to null. ■

If the decyphered code is unreachable, as we showed in Theorem 4.4, and we cannot encrypt the code for another CPU, according to Theorem 4.5, there still remains an alternative: to generate several computers with the same keys. Then the attacker shall have to pay only for one copy, and actually obtain several copies.

This attack cannot be prevented completely, since we must permit replacement of CPUs (see Section 3.3). Indeed the same problem exists in the other software protection methods, and the solutions available are usually rather unsatisfactory [HK84].

It is now proved that all the CPUs with the same keys, except one, should be returned to Z. Therefore the effect of these attacks is minimal. Given two computers with different keys, T17 must be used in order to make the keys of both computers equal and meaningful. Meaningful keys are keys that decypher programs distributed by T12 or T13.

THEOREM 3.6. *Let $S_0 = (M_0, X_0, K_0)$ be a state such that $M_0 = \emptyset$ and $X_0 = 0$ and all the keys are cryptographically independent. Then S is harmless, for $U = \{(M, X, K) \mid (j \neq i) (K_i = K_j = a^{-1} \neq \text{null}) \& (D_a a \in M) \& X < R\}$.*

Proof. Since $X < R$ in U , T17 cannot be used. The only transaction that changes keys is T15; but in order to use it, T16 must be employed. But if T16 has been used to produce $E_u(D_w; \text{replace})$, where $K_i = D_u$ and $K_j = D_w$ before T16, then $K_j = \text{null}$ after T16, and since T15 may be used only for C_i , S is still harmless for U . ■

We state somewhat unformally and without proof the following theorem, which finds the expenses of the attacker for obtaining n computers with identical keys.

THEOREM 3.7. *If S is harmless for $U = \{(M, X, K) \mid (X < R) \& (i \neq j) (K_i = K_j \neq \text{null})\}$ then it is harmless for $W = \{(M, X, K) \mid (X < R * \log_2(q)) \& (|I| = q) \& ((i, j \in I) \Rightarrow (K_i = K_j \neq \text{null}))\}$.*

The next result is, perhaps, of minor importance. We prove that even if T18 is used, and all the keys in a CPU are revealed, the attackers cannot forge the signature of Z. Thus the attackers still have to order software by sending the correct

public key. This result holds only when PPS is implemented using PKCS. We denote PPS/PKV to be PPS/PK with the addition of T18. Let V be the price for violating the integrity of the CPU.

THEOREM 3.8. *In PPS/PKV, if S is harmless for $U_1 = \{(M, X, K) | (D_z a \in M) \& (X < V)\}$ then it is harmless for $U_2 = \{(M, X, K) | (D_z a \in M)\}$.*

Proof. There is no transaction, including T18, that performs D_z on a given string. ■

4. PPS Implemented with Conventional Cryptosystem.

Implementing PPS by PKCS is quite natural, but also quite difficult. No chip available performs a PKCS, and the security of PKCS is still in doubt. Conventional cryptosystems are more mature. Several methods have been implemented in integrated circuits and are considered quite secure. Most known is [DES77].

The implementation of PPS by a conventional cryptosystem is based on emulating the required properties of PKCS by adding redundant information. Two features of PKCS are used in PPS:

- 1) Signatures - used to ensure that keys are not invented.
- 2) Secrecy - the program is encrypted by the distributor, yet he cannot decipher programs encrypted by other distributors.

4.1. PPS/C.

When using conventional cryptosystems, the signatures implemented with PKCS before, are now implemented by the processors. Each processor C_u contains three hidden keys:

- 1) K_z - The key of Z .
- 2) K_u - Computer's key
- 3) $F_u(i)$ - Temporal key for indirect software distribution

The emulation is performed by implementing E_z , D_z with conventional keys and the protocols are given in the following sections. Section 5.5 contains the corresponding transactions, which forms a TS denoted by PPS/C.

We assume the cryptosystems are secure, i.e. an attacker cannot determine m from $K_a(m)$, without knowing K_a . It is also impossible to find K_a from m and $K_a(m)$. Most cryptosystems are presumed to be secure in this manner. Note that we permit the encryption to be commutative, i.e. $K_a K_b(a) = K_b K_a(a)$.

4.2. Direct software distribution protocol (PPS/C).

The following is the protocol for direct distribution of software, from producer P to the user U. The words *key*, *prog* and *replace* are predefined strings used in the protocol. It is implicit that, whenever possible, honest participants in the protocol check for those strings in the input.

- D1. $(U, K_z(K_u; key), P)$ - The user sends key K_u signed and hidden by K_z .
- D2. $(P, K_z(K_u; key), PGM), C_p)$ - The producer enters both users' key and program into his computer...
- D3. $(C_p, K_u(PGM; prog), P)$ - The encrypted program is given to the producer.
- D4. $(P, K_u(PGM; prog), U)$ - The producer transfers the encrypted program to the user.
- D5. $(U, K_u(PGM; prog), C_u)$ - The user gives his computer the encrypted program.
- D6. $(C_u, O(PGM), U)$ - The computer executes the program.

4.3. Indirect software distribution protocol (PPS/C).

The following is the protocol for indirect distribution of software, from producer P to a user U via a dealer L.

- I1. $(P, F_u(i)[PGM, F_u(i+1)], L)$ - Producer P sells token i to dealer L. This step may be done (for several tokens) before the other steps of the

protocol.

- I2. $(U, K_z(K_u; key), L)$ - User U sends his public key to the dealer.
- I3. $(L, (K_z(K_u; key), F_u(i)[PGM, F_u(i+1)]), C_i)$ - The dealer uses token i .
- I4. $(C_i, K_u(PGM; prog), L)$ - The encrypted program is given to the dealer. In the same time, C_i changes from $F_u(i)$ to $F_u(i+1)$.
- I5. $(L, K_u(PGM; prog), U)$ - From this step - same as direct distribution.
- I6. $(U, K_u(PGM; prog), C_u)$ - The user enters the program into his computer.
- I7. $(C_u, O(PGM), U)$ - The computer executes the program.

4.4. CPU replacement protocol (PPS/C)

The following protocol in PPS/C is for the replacement of a users' CPU. The serviceperson S replaces C_u with C_s , by the help of Z .

- R1. $(U, K_z(K_u; key), S)$ - User U sends his key to S .
- R2. $(S, (K_z(K_u; key), K_z(K_s; key)), Z)$ - The serviceperson sends both keys to Z .
- R3. $(Z, K_s(K_u; replace), S)$ - Note that Z , in PPS/C, does not have to keep track of the keys.
- R4. $(S, K_s(K_u; replace), C_s)$ - New keys installation.
- R5. The CPUs are replaced. The replaced CPU ought to be returned to Z .

4.5. PPS/C as a TS

The transactions of PPS/C are listed in Table 2 for computers C_u , and C_w . The variables of PPS/C are: X is the total expenses of the attackers, for every user u , K_u is the key of his computer C_u . For every dealer L , Q_i is the temporal key $F_i(i)$. The set M of all the messages transmitted so far, which corresponds to the information held by the attackers.

Theorems 4.3-4.8 may be proved for PPS/C, but since they are simple and similar to the proofs for PPS/PK, we will not give them here.

5. Conclusion

The problem of software piracy causes considerable losses to software producers. The scheme presented - PPS - provides proved, reliable protection, and convenient protocols for distribution of software and replacement of CPUs. PPS requires implementation of cryptographic capabilities - public key or conventional key - inside the CPU. This is a challenge for all CPU manufacturers !

We believe that by using suitable protection methods software piracy could be rendered obsolete. Such a step will be to the benefit of all the parties involved (well, almost ...).

6. Acknowledgments

We thank Mr. Gadi Karmi for his proofreading.

7. References:

- [AM84] D.J. Albert and S.P. Morse, "Combating Software Piracy by Encryption and Key Management", *Computer* April 1984
- [DES77] National Bureau of Standard, "Data Encryption Standard", *Federal Information Processing Standard Publication 46*, January 1977
- [DH76] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, Vol. IT-22, 1976
- [HK84] A. Herzberg and G. Karmi, "On Software Protection" *Proc. Fourth JCIT, Jerusalem, Israel* April 1984
- [HP85] A. Herzberg and S. Pinter, "The Transaction System Model and Security Engineering", *in preparation*
- [M83] M.J. Merritt, "Cryptographic Protocols", *GIT-ICS-83/06, doctoral dissertation, The Georgia Institute of Technology*, 1983.
- [RSA78] R.L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and PKCs", *Comm. ACM*, Vol 21, No. 2 (Feb. 1978).

Table 1: Transactions of PPS/PK.

T#	Input	Output	Change	Steps
T1	$D_u E_u a$	a	—	D3
T2	$E_u D_u a$	a	—	—
T3	$E_u D_w a$	$D_w E_u a$	—	—
T4	$E_u E_w a$	$E_w E_u a$	—	—
T5	$D_u D_w a$	$D_w D_u b$	—	—
T6	$D_u E_w a$	$E_w D_u a$	—	—
T7	a, b	a(b)	—	—
T8	a	O(a)	—	—
T9	$E_u a$	O(a)	—	D5, I7
T10	—	$D_z E_u$	—	D1
T11	$D_z a, b$	a(b)	—	D2, D3
T12	$D_z a$	a(P)	X=X+cost	D2, D3
T13	$D_z a, F_u(i)[P, F_u(i+1)]$	a(P)	$Q_u = F_u(i+1)$	I3, I4
T14	—	$F_u(i)[P, F_u(i+1)]$	X=X+cost	I1
T15	$E_u(a; replace)$	—	$K_u = a$	R4
T16	$D_z E_u, D_z E_w$	$E_u(D_w; replace)$	$K_w = null$	R4, R5
T17	$D_z E_u, D_z E_w$	$E_u(D_w; replace)$	X=X+R	—
T18	—	D_u, E_z, Q_u	X=X+V	—

TABLE 2: : TRANSACTIONS OF PPS/C

T#	Input	Output	Change	Steps
T1	$K_u K_u a$	a	—	—
T2	$K_u K_w$	$K_w K_u$	—	—
T3	a, b	ab	—	—
T4	a	$Q(a)$	—	—
T5	$K_u(a, prog)$	$Q(a)$	—	D5, I7
T6	—	$K_s(K_u; key)$	—	D1, R1, R2, I2
T7	$K_s(a, key), b$	$a(b, prog)$	—	D3
T8	$K_s(a, key)$	$a(P, prog)$	$X = X + cost$	—
T9	$K_s(a, key), F_u(i)[P, F_u(i+1)]$	$a(P, prog)$	$Q_u = F_u(i+1)$	I3, I4
T10	—	$F_u(i)[P, F_u(i+1)]$	$X = X + ccst$	I1
T11	$K_u(a, replace)$	—	$K_u = a$	R4
T12	$K_s(K_u, key), K_s(K_w, key)$	$K_u(K_w, replace)$	$K_w = null$	R4, R5
T13	$K_s(K_u, key), K_s(K_w, key)$	$K_u(K_w, replace)$	$X = X + R$	—

Fingerprinting Long Forgiving Messages

G. R. Blakley

*C. Meadows**

G. B. Purdy

Department of Mathematics

Texas A&M University

College Station, Texas 77843-3368

In his 1983 paper, Neal Wagner¹ defines a *perfect fingerprint* to be an identifying fingerprint added to an object in such a way that any alteration to it that makes the fingerprint unrecognizable will also make the object unusable. A perfect fingerprinting scheme for binary data would seem difficult to devise, since it would be possible to discover the fingerprints by comparing different fingerprinted copies of the same piece of data. In this paper we discuss a fingerprinting scheme which, although it does not surmount this problem entirely, at least specifies the number of copies an opponent must obtain in order to erase the fingerprints.

The fingerprints involved will be rather lengthy, so we will restrict ourselves to what we will call *long forgiving messages*. A forgiving message is one which is still readily understandable and not jarring when up to 0.1% of it has been altered. Examples are voice and television. People can speak comfortably amid the noise of a cafeteria and can enjoy watching a television show with several pixels per frame altered. The idea in each case is that the support of the noise (the set outside which the additive noise must vanish) must have small

*Now at Computer Science and Systems Branch, Naval Research Laboratory, Washington, D. C. 20375.

measure. We must also require that our messages not be *too* forgiving, since otherwise it would be possible to erase the fingerprints by adding random noise and still have a usable message.

Let P be a long forgiving message (for example, a digital TV show). We wish to protect P from piracy by adding a different fingerprint F to each copy of P in such a way that a pirate who wishes to copy $P+F$ and distribute it illicitly cannot erase the information about the origin of P contained in F unless he has obtained a certain predetermined number of different copies. We will define a *d out of n fingerprint scheme* to be one in which n objects are fingerprinted, and in which the pirate must obtain d copies in order to erase the fingerprint from one copy. A fingerprint F must also obey the following constraints: If we think of the M -tuples P and F as functions from the set $\{1,2,\dots,M\}$ to \mathbb{Z}_2 then

- [a] $\text{Supp}(F)$, the subset of $\{1,2,\dots,M\}$ outside of which F vanishes, must be small enough so that F does not interfere with the viewability of the program.
- [b] $\text{Supp}(F)$ must be large enough so that F cannot be eradicated by random noise without affecting the usability of the message.

We construct the n fingerprints, F_1 through F_n in the following manner. Fix an integer k . For each subset A of $\{1,2,\dots,n\}$ of cardinality $\leq k$, choose a subset $S(A)$ of $\{1,2,\dots,M\}$ such that $A \neq B \Rightarrow S(A) \cap S(B) = \emptyset$. Then let

$$F_i = \sum_{i \in A} \chi(S(A))$$

where χ denotes the characteristic function. Note that n must be equal to

$$\sum_{j=1}^k (M_j)$$

and so k must be relatively small.

Suppose that the pirate has obtained copies 1 through l , with fingerprints F_1 through F_l , and that he wants to erase the fingerprint from $P + F_1$ by adding some function E to it so that the origin of $P + F_1 + E$ is not ascertainable by the owner. (We use the word *owner* to mean the owner of the pristine copy.) Ideally, of course, he would like E to be F_1 , but if not, at least he would like E to be the sum of some $\chi(S(A))$ s such that $1 \in A$ and some $\chi(S(A))$ s such that $1 \notin A$ (where \neg denotes the logical not). The former will serve to cancel out various components of F_1 , and the latter will serve to give the owner misleading information about the origins of the other copies that the pirate has obtained. Of course, the pirate would prefer, in order to give the most misleading information possible, to add on characteristic functions of sets $S(A)$ such that $A \cap X = \emptyset$. Since he has absolutely no way of finding out such an $S(A)$ (except by obtaining more copies), he is reduced to adding on some random function R if he wishes to do this. However, the support of R must be relatively small in order not to interfere with the usability of the message. It follows that the probability that the intersection of $\text{Supp}(R)$ and any $S(A)$ will be large enough to mislead the owner is small, and therefore that the addition of a random function will not be useful in hiding information.

Thus the pirate's best options are either to add on various $\chi(S(A))$ s that he knows or to add on functions whose supports are randomly chosen subsets of the $S(A)$ s. However, he usually cannot find out the various $S(A)$ s directly. What he *can* find are the sets at which the copies he possesses differ from each other. In particular, for each subset A of $X = \{1, 2, \dots, l\}$ of cardinality $\#(A) \leq l/2$, he can compute the set $B(A) =$

$$\{x \in Y \mid (P + F_i)(x) = (P + F_j)(x) \iff i, j \in A \text{ or } i, j \notin A\}.$$

(There is no point in computing $B(A)$ for $\#(A) > 1/2$, since $B(A) = B(X-A)$.) For example, $B(\{1\})$ is the set of all points at which $P + F_1$ differs from all other of the $P + F_i$ s. An element x would either be in $B(\{1\})$ because $x \in \text{Supp}(F_1)$ and $x \notin \text{Supp}(F_2)$ through $\text{Supp}(F_l)$, or because $x \notin \text{Supp}(F_1)$ but $x \in \text{Supp}(F_1)$ through $\text{Supp}(F_l)$. Similarly, $B(\{1,2\})$ is the set of all points at which $P + F_1$ and $P + F_2$ agree with each other but differ from the rest of the F_i s, and so on. Thus, if $1 \in A$, adding on $\chi(B(A))$ to $P + F_1$ is the same as changing $P + F_1$ at all points at which it agrees with the copies in A and disagrees with the copies not in A . For example, adding $\chi(B(\{1,2\}))$ to $P + F_1$ is the same as changing $P + F_1$ at all points at which it agrees with $P + F_2$ and disagrees with all the other copies the pirate possesses.

Lemma 1. *If $\#(A) < l - k$, then*

$$B(A) = \bigcup \{S(C) \mid \#(C) \leq k \text{ and } C \cap X = A\}$$

and if $\#(A) \geq l - k$, then

$$B(A) = \bigcup \{S(C) \mid \#(C) \leq k \text{ and } [C \cap X = A \text{ or } C \cap X = X - A]\}$$

where $\bigcup \{X \mid Y\}$ denotes the union of all sets X with property Y .

Proof. Suppose $x \in B(A)$. Then all the $P + F_i$ s such that $i \in A$ agree at x and disagree at x with all the $P + F_j$ s such that $j \in X - A$. Thus either

$$x \in \bigcap \{\text{Supp}(F_i) \mid i \in A\} = \bigcup \{S(C) \mid \#(C) \leq k \text{ and } C \cap X = A\}$$

or

$$\begin{aligned} x &\in \bigcap \{\text{Supp}(F_i) \mid i \in X - A\} \\ &= \bigcup \{S(C) \mid \#(C) \leq k \text{ and } C \cap X = X - A\}. \end{aligned}$$

Conversely, if x is in either of these sets, then all the $P + F_i$ s agree at x and disagree with all the $P + F_j$ s such that $j \in X - A$, and so $x \in B(A)$.

Thus the second part of the lemma follows. However, if $\#(A) > l-k$, then $\#(X-A) > k$, and thus there is no C of cardinality $\leq k$ such that $C \cap X = X-A$. Thus the first part of the lemma follows. **QED.**

The following corollary tells us that if the number of copies the pirate possesses is large enough, then he has enough information to erase the fingerprints entirely.

Corollary 2. *If $l > 2k$, then*

$$\bigcup \{B(A) \mid 1 \in A\} = \text{Supp}(F_1).$$

Proof. If $l > 2k$, then $\#(A) < l - k$ for all subsets A of $X = \{1, 2, \dots, l\}$ of cardinality $\leq l/2$. It follows from Lemma 1 that

$$B(A) = \bigcup \{S(C) \mid \#(C) \leq k \text{ and } C \cap X = A\}.$$

We thus have

$$\begin{aligned} \bigcup \{B(A) \mid 1 \in A \text{ and } A \subseteq X\} &= \\ \bigcup \{S(C) \mid 1 \in C \text{ and } \#(C) \leq k\} &= \text{Supp}(F_1). \end{aligned}$$

QED.

Thus in the case $l > 2k$, the pirate can determine F_1 and add it to $P + F_1$ in order to obtain a pristine copy P .

Suppose that $l \leq 2k$. We will show that in this case the pirate who seeks to obtain a pristine copy of P by adding on various $\chi(B(A))$ s not only cannot mislead the owner but risks giving him even more information than before.

Lemma 3. *Let $E = F_1 + Q$ where $\text{Supp}(Q)$ is the union of some set of $B(A)$ s. Suppose that there exists a t such that no $S(A)$ where $\#(A) \leq t$ appears in $\text{Supp}(E)$. If some $S(A)$ such that $\#(A) = t + 1$ appears in $\text{Supp}(E)$, then $A \subseteq X$.*

Proof. Suppose that $A \not\subseteq X$. Then $\#(A \cap X) \leq t$ and so $S(A \cap X)$ does not appear in $\text{Supp}(E)$ by hypothesis. By Lemma 1, we have $S(A) \subseteq B(A \cap X)$ and $S(A) \cap B(C) = \emptyset$ for any other $C \subseteq X$. Therefore

$B(A \cap X)$ appears in $\text{Supp}(E)$. We thus have only two possibilities. Either $1 \in A$ and $B(A \cap X)$ does not appear in $\text{Supp}(Q)$, or $1 \notin A$ and $B(A \cap X)$ does appear in $\text{Supp}(Q)$. In either case we have $S(A \cap X) \subseteq \text{Supp}(E)$, contradicting our assumption that no $S(C)$ where $\#(C) \leq t$ appears in $\text{Supp}(E)$. QED.

Lemma 4. *Suppose that $l \leq 2k$. Then the pirate who attempts to erase information about the origin of $P + F_1$ by adding to it various $\chi(B(A))$ s must add on all $\chi(B(A))$ such that $1 \in A$ and $\#(A) \leq \lfloor l/2 \rfloor$.*

Proof. The proof is by induction on the size of A . First, suppose that $A = \{1\}$. The pirate must add $\chi(B(\{1\}))$ to $P + F_1$. For he must remove $\chi(S(\{1\}))$ from $P + F_1$, since, if it were left in, the fact that $S(\{1\})$ is contained in $\text{Supp}(F_1)$ but in no other $\text{Supp}(F_i)$ would tell the owner that the pirate had had access to copy 1. But, since $S(\{1\})$ is contained in $B(\{1\})$ and no other $B(A)$, the pirate has no way of knowing which elements of $B(\{1\})$ are in $S(\{1\})$ and which aren't. Thus the only way the pirate can remove $S(\{1\})$ is by adding $\chi(B(\{1\}))$.

Next, assume that the pirate has added on all $\chi(B(A))$ for all A such that $1 \in A$ and $\#(A) \leq t$, for some $t < \lfloor l/2 \rfloor$. Let $E = F_1 + Q$, where Q is the function that the pirate has added on. We will show that the support of E contains no $S(A)$ such that $\#(A) \leq t$. Clearly, the pirate has erased all $\chi(S(A))$ such that $1 \in A \subseteq X$ and $\#(A) \leq t$. Moreover, he has not added on any $\chi(S(C))$ such that $\#(C) \leq t$. For by Lemma 1 the only way he could have done this would be if $X \cap C = X - A$, where A is one of the sets of cardinality $\leq t$ such that $\chi(B(A))$ was added on. But this would imply that $\#(X - A) \leq t$, and hence $l = \#(X) \leq 2t$, which contradicts our assumption that $t < \lfloor l/2 \rfloor$.

The owner can now conclude from Lemma 3 that if $\chi(S(A))$ appears in the support of E , and $\#(A) = t + 1$, then $A \subseteq X$. Moreover, such sets A exist, since $t + 1 \leq \lfloor l/2 \rfloor$ and $l \leq 2k$. Now all he has to do is

take the union of all A such that $\#(A) = t + 1$ and $\chi(S(A))$ appears in the support of E in order to find out which copies the pirate had access to.

Thus the pirate must do something further if he wants to hide the origin of his copy. He has two options. First, he can add on various $\chi(B(A))$ where $1 \in A$. But he can't add on any $\chi(B(\{a\}))$, or the owner will be able to tell, by the appearance of $S(\{a\})$ in the support of E , that the pirate had access to copy a . However, if he doesn't add on any $\chi(B(\{a\}))$, there is some $q \leq \lfloor l/2 \rfloor$ such that no $S(A)$ such that $\#(A) < q$ appears in the support E but some $S(A)$ such that $\#(A) = q$ does appear. The owner can then use Lemma 3 as before to find the other copies the pirate had access to.

The pirate's other option is to add on some or all of the $\chi(B(A))$ s such that $\#(A) = t + 1$ in order to erase some or all of such $\chi(S(A))$ s appearing in F_1 . But he must erase all such $\chi(S(A))$ s since if there was even one that he did not erase, the owner would again be able to conclude, using Lemma 3, that the pirate had had access to every copy a such that $a \in A$. **QED.**

Theorem 5. *Suppose that $l \leq 2k$. Then a pirate cannot erase information about the origin of $P + F_1$ by adding various $\chi(B(A))$ s without revealing information about the origins of the other copies he has access to.*

Proof. By Lemma 4 the pirate must add on all $\chi(B(A))$ such that $1 \in A$ and $\#(A) \leq \lfloor l/2 \rfloor$. It follows from Lemma 1 and the fact that $l \leq 2k$ that he has also added on all $\chi(S(X - A))$ such that $1 \in A \subseteq X$ and $\#(A) = \lfloor l/2 \rfloor$. In other words, he has added on all $\chi(S(A))$ such that $1 \in A \subseteq X$ and $\#(A) = l - \lfloor l/2 \rfloor$. Once again the owner can tell, from the absence of any $\chi(S(A))$ such that $\#(A) < l - \lfloor l/2 \rfloor$, that the owner has eliminated all such $\chi(S(A))$. Moreover, once again the owner can use Lemma 3 to can reason that, if any $\chi(S(A))$ such that

$\#(A) = 1 - [1/2]$ appears in the altered function, then $A \subseteq X$. The owner takes the union of all such A to find $X - \{1\}$. **QED.**

Thus if $2k \geq 1$, the pirate cannot erase information about the origin of $P + F_1$ by adding various $\chi(B(A))$ s to it without giving away information about the other copies he's obtained. But what if he adds on some $\chi(D)$ where D is a randomly chosen subset of some $B(A)$? If the pirate were lucky, such a D might contain all or most of the sets $S(C)$ such that $C \cap X = A$ and none or few of the sets $S(C)$ such that $C \cap X = X - A$. This can be made less likely by choosing the sets $S(C)$ large enough so that the chance that such a D would either miss any $S(C)$ entirely (if D is large) or contain an entire $S(C)$ (if D is small) or miss some $S(C)$ s and contain others, (if D is medium-sized) is negligible.

We are thus led to conclude that the fingerprint scheme described above is a $2k + 1$ out of n fingerprint scheme.

The construction of such a fingerprint scheme now seems easy. We simply choose the level of protection we desire and construct the appropriate sets $S(A)$. We are faced with one problem, however: the size of the fingerprints grows exponentially with the level of protection desired. As a matter of fact, since each fingerprint F_i is made up of all $S(A)$ such that $\#(A) \leq k$ and $i \in A$, we have, if $\#(S(A)) = s$ for each such A , that

$$\#(\text{Supp}(F_i)) = s \sum_{j=1}^{k-1} \binom{M}{j}$$

where M is the total number of messages. Thus the size of the fingerprints could easily grow to the point at which they start interfering with the messages.

We can get around the problem of exponential growth somewhat by using several less ambitious fingerprint schemes concurrently. For example, suppose that an owner wishes to protect about 27,000 copies of his message. If he used a 31 out of 27,000 fingerprint scheme, each

fingerprint would take up more than $s10^{50}$ bits. However, suppose that he constructs three 31 out of 31 fingerprint schemes $\{F_1, \dots, F_{31}\}$, $\{G_1, \dots, G_{31}\}$, and $\{H_1, \dots, H_{31}\}$. Each of these three fingerprint schemes takes up

$$s \sum_{j=1}^{14} \binom{31}{j} \approx s10^9$$

bits per fingerprint. Such fingerprint schemes will still take up a relatively small amount of space in something as large as a digital TV show. (If this number is still considered unmanageably large, the owner could instead construct, say, three 13 out of 31 fingerprint schemes, each of which would take up about $s200,000$ bits per fingerprint, as opposed to a 13 out of 27,000 fingerprint scheme, which would take up about 10^{20} bits per fingerprint.) The owner divides his distribution area into 31 geographic areas, each with 31 outlets selling 31 copies each. The i th copy in the j th outlet in the k th geographic area is fingerprinted by $F_i + G_j + H_k$. Thus if a pirate obtains all his copies from one outlet and attempts to erase the fingerprints we know exactly which copies he has obtained, if he obtains copies from different outlets in the same geographic area we no longer know exactly which copies they are, but we know the outlets they came from, and if he obtains copies from different geographical areas, we know the areas he visited, although we no longer know the individual outlets he obtained the copies from. Even in this last case, however, we still retain some information about the individual copies. Suppose, for example, that a pirate obtains copies $P + F_1 + G_1 + H_1$ and $P + F_2 + G_2 + H_2$. The owner who retrieves a tampered-with copy can determine that the pirate must have had access to at least two copies of the form $P + F_{i_1} + G_{j_1} + H_{k_1}$ and $P + F_{i_2} + G_{j_2} + H_{k_2}$ where $\{i_1, i_2\} = \{j_1, j_2\} = \{k_1, k_2\} = \{1, 2\}$, as shown in the following graph.

H_1				H_2			
G_1		G_2		G_1		G_2	
F_1	F_2	F_1	F_2	F_1	F_2	F_1	F_2

Thus he knows there are only eight possibilities for the origins of the two shows.

We have written largely in terms of discrete messages, i. e., messages with finitely many symbols taken from a finite alphabet. But it is clearly possible to do something analogous with continuous messages. A reader who deals with these matters can fill in the details in the obvious way.

NSA Grant MDA 904-83-H-0002 partially supported this work.

References

1. Neal Wagner, "Fingerprinting," *Proceedings of the 1983 Symposium on Security and Privacy*, pp. 18-22, IEEE Computer Society, Oakland, CA, April 25-27, 1983.

CRYPTANALYSIS OF DES WITH A REDUCED NUMBER OF ROUNDS

SEQUENCES OF LINEAR FACTORS IN BLOCK CIPHERS

David Chaum & Jan-Hendrik Evertse

Centre for Mathematics and Computer Science
Kruislaan 413 1098 SJ Amsterdam The Netherlands

1. INTRODUCTION

A blockcipher is said to have a linear factor if, for all plaintexts and keys, there is a fixed non-empty set of key bits whose simultaneous complementation leaves the exclusive-or sum of a fixed non-empty set of ciphertext bits unchanged.

Since it appears infeasible to test all possible combinations of key bits and ciphertext bits for DES [NBS 77], we tried to find linear structures in the separate rounds of DES and hoped that these structures could be combined to yield a linear factor over the whole cipher. This naturally led us to the notion of "sequences of linear factors." In general, there might be linear factors that cannot be derived from sequences of linear factors, but under our assumptions about DES (detailed below) it seems that factors for the whole cipher would consist of sequences of factors for the individual rounds. Our notion of sequence of linear factors extends that of "per round linear factors" introduced by Reeds and Manferdelli [84]. The essential difference is that sequences of linear factors allow different rounds to have different linear factors, while per round linear factors must remain the same for each round.

We have given several examples of blockciphers, consisting of consecutive rounds of DES, that are vulnerable to a known plaintext attack faster than exhaustive key search. For instance, the blockciphers consisting of the first 4, 5 or 6 rounds of DES can be attacked about 2^{19} , 2^9 , and 2^2 faster than by exhaustive key search, respectively. The results presented do not work for the blockcipher consisting of rounds 1-7 of DES, but for the blockcipher consisting of rounds 2-8 we can save a factor 2.

This research was supported in part by the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

H.C. Williams (Ed.): Advances in Cryptology - CRYPTO '85, LNCS 218, pp. 192-211, 1986.
© Springer-Verlag Berlin Heidelberg 1986

The attacks considered are of the "meet-in-the-middle" type. Such an attack on a blockcipher composed of R consecutive rounds of DES can be described as follows: Suppose a cryptanalyst has a plaintext p and corresponding ciphertext c . For each guessed key k the cryptanalyst enciphers p with the first S rounds of DES yielding d' , and deciphers c with the last $R - S$ rounds yielding d'' . If $d' = d''$, the cryptanalyst concludes that k is the true key. Considerably less guesses for the key are required compared to exhaustive key search when there are i and j such that both the j -th bit of d' and the j -th bit of d'' are independent of the i -th key bit. By independence we mean that for all p , c , and k , the j -th bit of d' and the j -th bit of d'' are unchanged when the i -th bit of k is complemented.

Meyer [78] argued that blockciphers consisting of R consecutive rounds of DES can have ciphertext bits independent of key bits if and only if $R \leq 4$. In his arguments he used the unproved assumption that between two adjacent rounds of DES no dependencies are cancelled. This assumption means that if some output bit of the i -th round is functionally dependent on certain input bits for the i -th round and if some of these input bits are functionally dependent on the i -th key bit, then that output bit is also dependent on the i -th key bit. Meyer's assumption can be considered as a special case of the assumption that linear factors in DES always result from sequences of linear factors in the individual rounds. Under this general assumption, we show that blockciphers consisting of eight or more consecutive rounds of DES have no linear factors, and as a special case, that such ciphers are not subject to the kind of meet-in-the-middle attacks described above.

The next section explains how linear structures can be helpful in cryptanalysis while introducing some necessary notation. Subsequent sections consider whether DES with a reduced number of rounds has such structures. Potential extensions to more rounds of DES are mentioned in our concluding remarks.

2. LINEAR STRUCTURES IN BLOCKCIPHERS

This section gives an overview of various kinds of linear structures which blockciphers can have, together with their possible consequences for cryptanalysis. Some of the ideas in this section are included in [Hellman et al 76] and [Reeds and Manferdelli 84].

Some elementary notation that will allow us to make precise statements in the remainder of the paper is now introduced. Let $\mathbb{F}_2 = \{0, 1\}$ be the finite field of two elements. By \mathbb{F}_2^n we shall denote the vector space of n -tuples over \mathbb{F}_2 . Elements of \mathbb{F}_2^n are denoted by bold characters such as \mathbf{x} or strings $x_1 x_2 \cdots x_n$, with $x_i \in \mathbb{F}_2$, and with the coordinates of \mathbf{x} commonly referred to as "bits." Elements of the cartesian product $\mathbb{F}_2^{n_1} \times \cdots \times \mathbb{F}_2^{n_r}$ are often denoted by (x_1, \cdots, x_r) , where $x_i \in \mathbb{F}_2^{n_i}$ for $i = 1, \cdots, r$. When using notions from linear algebra, such as vectors, vector spaces, bases, linear mappings, etc. we assume that the underlying field of scalars is \mathbb{F}_2 . In particular, the $+$ sign denotes addition of vectors over \mathbb{F}_2 , sometimes referred to as exclusive-or. If A is a linear mapping, then $\text{im}(A)$ and $\text{ker}(A)$ will denote its image and null space, respectively.

If \mathcal{U} is a vector space and if $\mathcal{U}_1, \cdots, \mathcal{U}_r$ are subspaces of \mathcal{U} , then $\sum_{j=1}^r \mathcal{U}_j$ denotes the smallest

subspace of \mathcal{U} containing $\mathcal{U}_1, \dots, \mathcal{U}_r$. The subspace of \mathcal{U} generated by the set $\{\mathbf{x}_\alpha: \alpha \in A\}$ is denoted by $\{\mathbf{x}_\alpha: \alpha \in A\}$.

By a blockcipher, we mean a mapping $F: \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{M}$, where $\mathcal{M} = \mathbb{F}_2^m, \mathcal{K} = \mathbb{F}_2^r$ are the message space and key space respectively, such that for each $\mathbf{k} \in \mathcal{K}$, the mapping $F(\cdot, \mathbf{k}): \mathcal{M} \rightarrow \mathcal{M}$ is invertible. We denote decryption by $F^{-1}: \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{M}$, i.e. if $\mathbf{c} \in \mathcal{M}, \mathbf{k} \in \mathcal{K}$ then $F^{-1}(\mathbf{c}, \mathbf{k})$ is equal to \mathbf{p} , where \mathbf{p} is the element of \mathcal{M} for which $F(\mathbf{p}, \mathbf{k}) = \mathbf{c}$. Finally, if $F_1, \dots, F_R: \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{M}$ are blockciphers, then the product $F = F_R F_{R-1} \dots F_1$ of F_1, \dots, F_R is defined as follows: if $\mathbf{p} \in \mathcal{M}, \mathbf{k} \in \mathcal{K}$, and if the sequence $\mathbf{p}_1, \mathbf{p}_2, \dots$ is defined by

$$\mathbf{p}_1 = F_1(\mathbf{p}, \mathbf{k}), \mathbf{p}_2 = F_2(\mathbf{p}_1, \mathbf{k}), \dots, \mathbf{p}_r = F_r(\mathbf{p}_{r-1}, \mathbf{k}),$$

then

$$F(\mathbf{p}, \mathbf{k}) = \mathbf{p}_r.$$

Let $F: \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{M}$ be a blockcipher, where $\mathcal{M} = \mathbb{F}_2^m, \mathcal{K} = \mathbb{F}_2^r$. If $\mathbf{c} = F(\mathbf{p}, \mathbf{k})$ with $\mathbf{p} = p_1 \dots p_m, \mathbf{k} = k_1 \dots k_r, \mathbf{c} = c_1 \dots c_m$, then

$$c_i = f_i(p_1, \dots, p_m, k_1, \dots, k_r) \text{ for } i = 1, \dots, m, \quad (1)$$

where the $f_i: \mathbb{F}_2^{m+r} \rightarrow \mathbb{F}_2$ are boolean functions. Suppose that there are sets $\{i_1, \dots, i_s\} \subseteq \{1, \dots, m\}, \{j_1, \dots, j_r\} \subseteq \{1, \dots, r\}$ such that the functions f_{i_s} are independent of the key bits k_i with i different from i_1, \dots, i_s , that is

$$c_{i_s} = f_{i_s}(p_1, \dots, p_m, k_{i_1}, \dots, k_{i_s}) \text{ for } k = 1, \dots, r.$$

This can be written more conveniently as

$$\tilde{\mathbf{c}} = \tilde{F}(\mathbf{p}, \tilde{\mathbf{k}}), \quad (2)$$

where $\tilde{\mathbf{k}} = k_{i_1} \dots k_{i_s}, \tilde{\mathbf{c}} = c_{j_1} \dots c_{j_r}, \tilde{F}: \mathbb{F}_2^m \times \mathbb{F}_2^s \rightarrow \mathbb{F}_2^r$.

Suppose that a cryptanalyst knows a pair of plaintext and corresponding ciphertext (\mathbf{p}, \mathbf{c}) of F and wants to find the key \mathbf{k} from the equation

$$\mathbf{c} = F(\mathbf{p}, \mathbf{k}). \quad (3)$$

The cryptanalyst may use the following method: (i) solve $\tilde{\mathbf{k}} = k_{i_1} \dots k_{i_s}$ from (2) by exhaustively trying all $\tilde{\mathbf{k}}$ (a value of $\tilde{\mathbf{k}}$ can be tried by extending $\tilde{\mathbf{k}}$ to a key \mathbf{k} by setting all key bits k_i with i different from i_1, \dots, i_s to zero, computing $F(\mathbf{p}, \mathbf{k})$ and checking if the correct value of $\tilde{\mathbf{c}}$ appears) and (ii) solve \mathbf{k} from (3) by exhaustively trying all \mathbf{k} for which k_{i_1}, \dots, k_{i_s} are equal to

the values found in (i). Assuming that in step (i) only one solution is found, the cryptanalyst has to do about

$$2^s + 2^{n-s} \quad (4)$$

computations of F before finding the key. In general, we may not assume that only one solution is found in (i). The number of solutions found in (i) can be reduced if the cryptanalyst possesses M pairs of corresponding plaintext and ciphertext. Suppose that the cryptanalyst has the plaintext-ciphertext pairs $(p_1, c_1), \dots, (p_M, c_M)$ and wants to solve the key from

$$c_1 = F(p_1, k), \dots, c_M = F(p_M, k). \quad (5)$$

Instead of doing (i),(ii), the following method may be used:

(i'): try all values for \tilde{k} . If a \tilde{k} is found with $\tilde{F}(p_1, \tilde{k}) = \tilde{c}_1$, then check if $\tilde{F}(p_2, \tilde{k}) = \tilde{c}_2$, $\tilde{F}(p_3, \tilde{k}) = \tilde{c}_3, \dots$ until some i is found with $\tilde{F}(p_i, \tilde{k}) \neq \tilde{c}_i$ or $i = M$. Accept \tilde{k} as a solution if $\tilde{F}(p_i, \tilde{k}) = \tilde{c}_i$ for $i = 1, \dots, M$;

(ii'): try all values for k for which $k_{i_1} \dots k_{i_s}$ is equal to one of the accepted solutions in (i'). If $F(p_1, k) = c_1$ then check if $F(p_2, k) = c_2, F(p_3, k) = c_3, \dots$ until $F(p_i, k) \neq c_i$ for some i or $i = M$. Accept k as a correct key if $F(p_i, k) = c_i$ for $i = 1, \dots, M$.

This algorithm finds all keys k with $F(p_i, k) = c_i$ for $i = 1, \dots, M$. In order to estimate the expected number of encipherments needed in steps (i'),(ii') we make the following very heuristic assumptions: for $k = 1, \dots, r, i = 1, \dots, M$ and for all wrong values of $\tilde{k} = k_{i_1} \dots k_{i_s}$ the $f_{j_i}(p_i, \tilde{k})$ are mutually independent uniformly distributed random variables on $\{0, 1\}$; for all j different from j_1, \dots, j_r , for all i in $\{1, \dots, M\}$ and all wrong values of k the $f_j(p_i, k)$ are also mutually independent uniformly distributed random variables on $\{0, 1\}$. With these assumptions, the expected number of encipherments in step (i') is

$$M + (2^s - 1)(1 - 2^{-r}) \times (1 + 2 \times 2^{-r} + 3 \times 2^{-2r} + \dots) \\ \leq M + \frac{2^s}{1 - 2^{-r}}$$

(where the term M comes from the correct value of \tilde{k}). The expected number of keys which have to be tried in step (ii') is equal to the product of 2^{n-s} and the expected number of accepted values for \tilde{k} in step (i'), namely

$$2^{n-s} \left[1 + (2^s - 1)2^{-Mr} \right].$$

Hence the expected number of encipherments in step (ii') is at most

$$\begin{aligned}
& M + \left[2^{n-s} \left(1 + (2^s - 1)2^{-Mr} \right) - 1 \right] (1 - 2^{r-m}) \times \\
& \quad \times \left[1 + 2 \times 2^{r-m} + 3 \times 2^{2(r-m)} + \dots \right] \\
& \leq M + \frac{2^{n-s} + 2^{n-Mr}}{1 - 2^{r-m}}.
\end{aligned}$$

Therefore, the expected total number of encipherments is at most

$$2M + \frac{2^s}{1 - 2^{-r}} + \frac{2^{n-s} + 2^{n-Mr}}{1 - 2^{r-m}}. \quad (6)$$

If $Mr > s$ this is only slightly larger than $2^s + 2^{n-s}$.

Suppose that $G, H: \mathfrak{M} \times \mathfrak{K} \rightarrow \mathfrak{M}$ are two blockciphers and that $F = HG$. Suppose that a cryptanalyst knows a plaintext-ciphertext pair of F , (\mathbf{p}, \mathbf{c}) say. Instead of solving the unknown key \mathbf{k} from (3), a cryptanalyst can try to solve \mathbf{k} from

$$G(\mathbf{p}, \mathbf{k}) = H^{-1}(\mathbf{c}, \mathbf{k}). \quad (7)$$

Attacks in which \mathbf{k} is solved from eq. (7) instead of (3) are called *meet-in-the-middle attacks*. Let $\mathbf{d}' = d'_1 \dots d'_m = G(\mathbf{p}, \mathbf{k})$, $\mathbf{d}'' = d''_1 \dots d''_m = H^{-1}(\mathbf{c}, \mathbf{k})$. Suppose that there are subsets $\{i_1, \dots, i_s\}$ of $\{1, \dots, n\}$ and $\{j_1, \dots, j_r\}$ of $\{1, \dots, m\}$ such that $d'_{j_1}, \dots, d'_{j_r}, d''_{j_1}, \dots, d''_{j_r}$ are functionally independent of the key bits k_i with i different from i_1, \dots, i_s . In other words, there are boolean functions $g_1, \dots, g_r, h_1, \dots, h_r$ such that

$$\begin{aligned}
d'_{j_1} &= g_1(\mathbf{p}, k_{i_1}, \dots, k_{i_s}) & d''_{j_1} &= h_1(\mathbf{c}, k_{i_1}, \dots, k_{i_s}) \\
&\vdots & & \vdots \\
&\vdots & & \vdots \\
d'_{j_r} &= g_r(\mathbf{p}, k_{i_1}, \dots, k_{i_s}) & d''_{j_r} &= h_r(\mathbf{c}, k_{i_1}, \dots, k_{i_s})
\end{aligned} \quad (8)$$

Now the unknown key \mathbf{k} can be found by first solving k_{i_1}, \dots, k_{i_s} from $g_1 = h_1, \dots, g_r = h_r$ and then solving the remaining key bits from (3) or (7). If the cryptanalyst has M plaintext-ciphertext pairs, the number of G, H^{-1} computations needed is given by (6).

We now consider linear structures more general than independencies of ciphertext bits or "bits in the middle" from key bits. Suppose that $A: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^r, B: \mathbb{F}_2^m \rightarrow \mathbb{F}_2^r$ are surjective linear mappings and that there exists a function \tilde{F} such that

$$AF(\mathbf{p}, \mathbf{k}) = \tilde{F}(\mathbf{p}, B\mathbf{k}) \text{ for } \mathbf{p} \in \mathfrak{M}, \mathbf{k} \in \mathfrak{K}. \quad (9)$$

Given a known plaintext-ciphertext pair (\mathbf{p}, \mathbf{c}) it is possible to solve the unknown key \mathbf{k} from $\mathbf{c} = F(\mathbf{p}, \mathbf{k})$ by firstly solving $\tilde{\mathbf{k}}$ from $A\mathbf{c} = \tilde{F}(\mathbf{p}, \tilde{\mathbf{k}})$ and secondly solving \mathbf{k} from $\mathbf{c} = F(\mathbf{p}, \mathbf{k})$, under the

restriction that $Ak = \tilde{k}$. Using that $\text{im}(A)$ has cardinality 2^s while the equation $Ak = \tilde{k}$ has 2^{n-s} solutions, a cryptanalyst having M plaintext-ciphertext pairs can find the key in a number of encipherments which is given by (6).

The linear structures which can be used in a meet-in-the-middle attack are more general than those explained above. Such structures exist if there are blockciphers G, H with $F = HG$, surjective linear mappings $A: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^r, B: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^s$ and functions \tilde{G}, \tilde{H} , such that for $p, c \in \mathcal{M}, k \in \mathcal{K}$,

$$AG(p, k) = \tilde{G}(p, Bk), \quad AH^{-1}(c, k) = \tilde{H}(c, Bk). \quad (10)$$

As mentioned in the introduction, given a blockcipher F , it might be infeasible to find out if it has any linear factors. Instead of this, one might try to represent F as a product of cryptographically weak blockciphers and check if these weak blockciphers themselves have such linear structures. Suppose that $F_1, \dots, F_R: \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{M}$ are blockciphers and that $F = F_R \cdots F_1$. Let A_i ($i = 0, \dots, R$) be linear mappings on \mathcal{M} and let B be a linear mapping on \mathcal{K} . We call $(A_0, \dots, A_R; B)$ a *sequence of linear factors* for F (with respect to F_1, \dots, F_R) if there are functions \tilde{F}_i ($i = 0, \dots, R$): $\text{im}(A_{i-1}) \times \text{im}(B) \rightarrow \text{im}(A_i)$ such that for $p \in \mathcal{M}, k \in \mathcal{K}$,

$$A_i F_i(p, k) = \tilde{F}_i(A_{i-1} p, Bk) \quad \text{for } i = 1, \dots, R. \quad (11)$$

Then there is a function $\tilde{F}: \text{im}(A_0) \times \text{im}(B) \rightarrow \text{im}(A_R)$ such that

$$A_R F(p, k) = \tilde{F}(A_0 p, Bk). \quad (12)$$

Note again that there may be linear mappings A_0, A_R, B satisfying (12) for some \tilde{F} which do not belong to sequences of linear factors.

Let $G = F_M F_{M-1} \cdots F_1, H = F_R F_{R-1} \cdots F_{M+1}$. Then $F = HG$. In a meet-in-the-middle attack we will need sequences of linear factors $(A_0, \dots, A_M; B), (A'_R, A'_{R-1}, \dots, A'_M; B)$ for G, H^{-1} respectively, such that

$$A_M = A'_M.$$

Also note that if $(A'_R, \dots, A'_M; B)$ is a sequence of linear factors for H^{-1} then $(A'_M, \dots, A'_R; B)$ is not necessarily a sequence of linear factors for H .

We need no longer distinguish between sequences of linear factors $(A_0, \dots, A_R; B)$, $(A'_0, \dots, A'_R; B')$ with $\ker(A_i) = \ker(A'_i)$ ($i = 0, \dots, R$), $\ker(B) = \ker(B')$, since they give us exactly the same advantage in finding the key. Thus we are mainly interested in sequences of vector spaces $(\mathcal{V}_0, \mathcal{V}_1, \dots, \mathcal{V}_R; \mathcal{W})$ where $\mathcal{V}_i = \ker(A_i), \mathcal{W} = \ker(B)$ for some sequence of linear factors $(A_0, \dots, A_R; B)$. Such sequences of vector spaces are called *sequences of factor spaces* for F . The following lemma characterizes these sequences of factor spaces.

Lemma 1. Let $\mathcal{V}_0, \dots, \mathcal{V}_R \subseteq \mathcal{M}$, $\mathcal{W} \subseteq \mathcal{K}$ be vector spaces. Then the following statements are equivalent.

(i) $(\mathcal{V}_0, \dots, \mathcal{V}_R; \mathcal{W})$ is a sequence of factor spaces for F .

(ii) $F_i(\mathbf{p} + \mathbf{x}, \mathbf{k} + \mathbf{y}) + F_i(\mathbf{p}, \mathbf{k}) \in \mathcal{V}_i$

for all $i \in \{1, \dots, R\}$, $\mathbf{p} \in \mathcal{M}$, $\mathbf{k} \in \mathcal{K}$, $\mathbf{x} \in \mathcal{V}_{i-1}$, $\mathbf{y} \in \mathcal{W}$.

Proof: (i) \rightarrow (ii). Let $(A_0, \dots, A_R; B)$ be a sequence of linear factors for F with $\ker(A_i) = \mathcal{V}_i$ ($i = 0, \dots, R$), $\ker(B) = \mathcal{W}$. It is easy to check, that for $i = 1, \dots, R$, $\mathbf{p} \in \mathcal{M}$, $\mathbf{k} \in \mathcal{K}$, $\mathbf{x} \in \mathcal{V}_{i-1}$, $\mathbf{y} \in \mathcal{W}$,

$$A_i F_i(\mathbf{p} + \mathbf{x}, \mathbf{k} + \mathbf{y}) = A_i F_i(\mathbf{p}, \mathbf{k}).$$

This proves (ii).

(ii) \rightarrow (i). Choose linear mappings A_0, \dots, A_R, B such that $\ker(A_i) = \mathcal{V}_i$ ($i = 0, \dots, R$), $\ker(B) = \mathcal{W}$. Define functions $\tilde{F}_i: \text{im}(A_{i-1}) \times \text{im}(B) \rightarrow \text{im}(A_i)$ ($i = 1, \dots, R$) as follows: if $\tilde{\mathbf{p}} \in \text{im}(A_{i-1})$, $\tilde{\mathbf{k}} \in \text{im}(B)$ then choose \mathbf{p}, \mathbf{k} such that $A_{i-1}\mathbf{p} = \tilde{\mathbf{p}}$, $B\mathbf{k} = \tilde{\mathbf{k}}$ and put $\tilde{F}_i(\tilde{\mathbf{p}}, \tilde{\mathbf{k}}) = A_i F_i(\mathbf{p}, \mathbf{k})$. From statement (ii) it follows that the \tilde{F}_i are well-defined (i.e. independent of the choice of \mathbf{p}, \mathbf{k}) and that for $\mathbf{p} \in \mathcal{M}$, $\mathbf{k} \in \mathcal{K}$,

$$A_i F_i(\mathbf{p}, \mathbf{k}) = \tilde{F}_i(A_{i-1}\mathbf{p}, B\mathbf{k}).$$

This proves (i). \square

2.1. SOME GENERALIZATIONS

Here we briefly mention some ways in which sequences of linear factors can be generalized. We have not looked for such general structures in DES. As before, F_1, \dots, F_R are blockciphers and $F = F_R F_{R-1} \dots F_1$. One possibility is to consider sequences of factors $(A_0, \dots, A_R; B)$ where $A_0, \dots, A_R; B$ are not necessarily linear mappings satisfying (11) for certain functions \tilde{F}_i . Such sequences can be helpful in cryptanalysis if B is a simple mapping, such as a linear mapping, a mapping composed of low degree polynomials over \mathbb{F}_2 , etc.

A second generalization considers sequences of *near* linear factors. This notion is an extension of an idea presented in [Hellman et al 76]. A sequence of linear mappings $(A_0, \dots, A_R; B)$ is called a sequence of near linear factors for F valid for a set \mathcal{S} of pairs of plaintexts and keys if there are functions \tilde{F}_i such that for each pair (\mathbf{p}, \mathbf{k}) in \mathcal{S} and each i with $1 \leq i \leq R$,

$$A_i \mathbf{p}_i = \tilde{F}_i(A_{i-1} \mathbf{p}_{i-1}, B \mathbf{k}),$$

where $\mathbf{p}_0 = \mathbf{p}$, $\mathbf{p}_i = F_i(\mathbf{p}_{i-1}, \mathbf{k})$. Suppose that F has a sequence of near linear factors $(A_0, \dots, A_R; B)$ valid for a set \mathcal{S} containing pairs (\mathbf{p}, \mathbf{k}) for each key \mathbf{k} or more generally, that F has sequences of linear factors $(A_0, \dots, A_R; B)$, all having the same A_0, A_R, B and valid for sets

S_1, \dots, S_r respectively, such that $S_1 \cup \dots \cup S_r$ contains pairs (p, k) for each k . Then there exist a positive number C and a function \tilde{F} such that for each key k , the relations

$$A_R F(p, k) = \tilde{F}(A_0 p, Bk) \quad (13)$$

are valid for a fraction $\frac{1}{C}$ of the plaintexts p . If a cryptanalyst has C pairs of corresponding plaintext and ciphertext, then for each pair (p, c) the key can be solved, under the hypothesis that (13) holds for the plaintext p . Thus C keys are found, one of which is expected to be the correct key.

A blockcipher F is said to have key clustering if there exist a mapping \tilde{F} and a non-injective linear mapping B such that for each key k , the relation

$$F(p, k) = \tilde{F}(p, Bk)$$

holds for a positive fraction of the plaintexts p . Desmedt, Quisquater and Davio [84] gave a few examples of key clustering in blockciphers consisting of at most three rounds of DES. The method by which these examples have been constructed can be described in terms of sequences of near linear factors as mentioned above.

3. MEET-IN-THE-MIDDLE ATTACKS ON DES

Independencies of "bits in the middle" from key bits in DES, which can be helpful in a meet-in-the-middle attack, are the subject of this section. First we give an overview of the mappings used in DES, assuming that the reader is familiar with the NBS description of the Data Encryption Standard. (For the complete description, we refer to [NBS 77]). In this paper, we use a slightly modified version of DES in which $IP, IP^{-1}, PC1$ are not used and E, P are combined to one table EP (cf. Davio et al [83], pp. 184-185). Thus the following mappings are used in our version of DES:

$EP: \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{48}$: EPx is formed from x as follows: first $y = Px$ is formed by permuting the 32 bits of x ; then $EPx = Ey$ is formed by taking 16 of the 32 bits of y once and the other 16 twice;

$S_j: \mathbb{F}_2^6 \rightarrow \mathbb{F}_2^4$ ($j = 1, \dots, 8$): the mappings defined by the S-boxes;

$S: \mathbb{F}_2^{48} \rightarrow \mathbb{F}_2^{32}$: $S(x) = (S_1 x_1, \dots, S_8 x_8)$ for $x = (x_1, \dots, x_8)$ with $x_j \in \mathbb{F}_2^6$;

$L_i: \mathbb{F}_2^{56} \rightarrow \mathbb{F}_2^{48}$ ($i = 1, \dots, 16$): $L_i k = PC2(C^{r(i)} k_1, C^{r(i)} k_2)$ for $k = (k_1, k_2)$ with $k_1, k_2 \in \mathbb{F}_2^{28}$. Here Cx is formed from x by applying a cyclic left shift to the bits of x , $r(i)$ is an integer determined by the shift pattern in the NBS-description of the key-scheduling and $PC2(x, y)$ is formed from x, y by selecting 24 bits from x , selecting 24 bits from y and permuting the selected 48 bits in some order.

The mappings EP, L_i are linear. If A is a linear mapping, then we say that A sends p to q if A maps the vector of which only the p -th bit is equal to 1 onto a vector of which at least the q -th

bit equals 1. If A maps the vector with only a 1 in its p -th bit onto 0, we say that A does not choose p . Thus EP sends each p in $\{1, \dots, 32\}$ to either one or two elements of $\{1, \dots, 48\}$, while each p in $\{1, \dots, 56\}$ is either not chosen by L_i or sent to exactly one element of $\{1, \dots, 48\}$.

We shall now algebraically describe our version of DES. The message space is \mathbb{F}_2^{64} . Elements of \mathbb{F}_2^{64} will be written in the form (x, y) , where $x, y \in \mathbb{F}_2^{32}$. The key space is \mathbb{F}_2^{56} .

The mappings $F_i: \mathbb{F}_2^{64} \times \mathbb{F}_2^{56} \rightarrow \mathbb{F}_2^{64}$ ($i = 1, \dots, 16$) (the "rounds") are defined by

$$F_i(q_0, q_1, k) = (q_1, q_0 + S(EPq_1 + L_i k)) \quad \text{for } (q_0, q_1) \in \mathbb{F}_2^{64}, k \in \mathbb{F}_2^{56}$$

and $DES: \mathbb{F}_2^{64} \times \mathbb{F}_2^{56} \rightarrow \mathbb{F}_2^{64}$ is defined by

$$DES = F_{16} F_{15} \cdots F_1.$$

Thus if q_2, q_3, \dots are defined by the recurrence sequence

$$q_{i+1} = q_i + S(EPq_i + L_i k) \quad (i = 1, \dots, 16)$$

then $DES(q_0, q_1, k) = (q_{16}, q_{17})$.

Let R, T be integers with $1 \leq R \leq T \leq 16$. We define

$$DES_{R,T} = F_T F_{T-1} \cdots F_R,$$

$$DES_{R,T}^{-1} = F_R^{-1} F_{R+1}^{-1} \cdots F_T^{-1}.$$

Let R, M, T be integers with $1 \leq R \leq M \leq T \leq 16$. For $p, c \in \mathbb{F}_2^{64}, k \in \mathbb{F}_2^{56}$, we put

$$\left. \begin{aligned} d' &= d'_1 \cdots d'_{64} = DES_{R,M}(p, k), \\ d'' &= d''_1 \cdots d''_{64} = DES_{M+1,T}^{-1}(c, k), \\ k &= k_1 \cdots k_{56}. \end{aligned} \right\} \quad (14)$$

Our aim is to find out if there are subsets $\{i_1, \dots, i_{64}\}, \{j_1, \dots, j_r\}$ of $\{1, \dots, 56\}, \{1, \dots, 64\}$ respectively, such that $d'_{j_1}, \dots, d'_{j_r}, d''_{j_1}, \dots, d''_{j_r}$ are functionally independent of the key bits k_i with i different from i_1, \dots, i_s .

Let p, c have the same meaning as above and put $p = (q_{R-1}, q_R), c = (q'_T, q'_{T+1})$. Define the sequences $q_{R-1}, q_R, q_{R+1}, \dots, q'_{T+1}, q'_T, q'_{T-1}, \dots$ by

$$q_{i+1} = q_i + S(EPq_i + L_i k) \quad (i = R, R+1, \dots),$$

$$q'_{i-1} = q'_i + S(EPq'_i + L_i k) \quad (i = T, T-1, \dots)$$

Let $t \in \{1, \dots, 56\}$. We define the sets $X_i(t)$ ($i = R-1, R, R+1, \dots$), $X'_i(t)$ ($i = T+1, T, T-1, \dots$) recursively as follows:

$$\left. \begin{aligned} X_{R-1}(t) &= X_R(t) = \emptyset ; X'_{T+1}(t) = X'_T(t) = \emptyset ; \\ X_{i+1}(t) &\text{ is the set of indices of the bits of } \mathbf{q}_{i+1} \text{ which functionally depend on} \\ &\text{some of the bits of } \mathbf{q}_i \text{ with indices in } X_i(t), \text{ some of the bits of } \mathbf{q}_{i-1} \text{ with} \\ &\text{indices in } X_{i-1}(t) \text{ and eventually } k_i; \\ X'_{i-1} &\text{ is defined similarly in terms of } X'_i, X'_{i-1}, k_i. \end{aligned} \right\} \quad (15)$$

Obviously, the sets of bits of $\mathbf{q}_i, \mathbf{q}'_i$ respectively, which are functionally dependent on k_i are included in $X_i(t), X'_i(t)$, respectively. An equivalent formulation of Meyer's assumption mentioned in §1 is that *all* indices in $X_i(t), X'_i(t)$ are of bits of \mathbf{q}, \mathbf{q}' , which are functionally dependent on k_i . For each t , we shall recursively compute the sets $X_i(t), X'_i(t)$. To this end, we introduce the following sets:

$$U_1 = \{1, 2, \dots, 6\}, U_2 = \{7, \dots, 12\}, \dots, U_8 = \{43, \dots, 48\}, \\ V_1 = \{1, 2, 3, 4\}, V_2 = \{5, \dots, 8\}, \dots, V_8 = \{29, \dots, 32\}.$$

Put $W_i(t) = \emptyset$ if L_i does not choose t and $W_i(t) = \{j\}$ if L_i sends t to an element of U_j . Finally, let \mathcal{F} be a function, mapping subsets of $\{1, \dots, 8\}$ onto subsets of $\{1, \dots, 8\}$ which is defined as follows: $\mathcal{F}(A)$ is the set of integers j with the property that there is an $i \in A$ such that EP sends an element of V_i to an element of U_j . In particular, $\mathcal{F}(\emptyset) = \emptyset$.

$\mathcal{F}(\{1\}) = \{2, 3, 4, 5, 6, 8\}$	$\mathcal{F}(\{5\}) = \{1, 2, 3, 4, 6, 7\}$	$\mathcal{F}(\emptyset) = \emptyset$ $\mathcal{F}(A \cup B) = \mathcal{F}(A) \cup \mathcal{F}(B)$
$\mathcal{F}(\{2\}) = \{1, 3, 4, 5, 7, 8\}$	$\mathcal{F}(\{6\}) = \{1, 2, 3, 5, 7, 8\}$	
$\mathcal{F}(\{3\}) = \{2, 4, 5, 6, 7, 8\}$	$\mathcal{F}(\{7\}) = \{1, 2, 3, 4, 6, 8\}$	
$\mathcal{F}(\{4\}) = \{1, 3, 5, 6, 7, 8\}$	$\mathcal{F}(\{8\}) = \{1, 2, 4, 5, 6, 7\}$	

table 1: the function \mathcal{F}

We define the sets $V_{R-1}(t), V_R(t), V_{R+1}(t), \dots, V'_{T+1}(t), V'_T(t), V'_{T-1}(t), \dots$ recursively as follows:

$$\left. \begin{aligned} V_{R-1}(t) &= V_R(t) = \emptyset, V'_{T+1}(t) = V'_T(t) = \emptyset, \\ V_{i+1}(t) &= V_{i-1}(t) \cup \mathcal{F}(V_i(t)) \cup W_i(t) \quad (i = R, R+1, \dots), \\ V'_{i-1}(t) &= V'_{i+1}(t) \cup \mathcal{F}(V'_i(t)) \cup W'_i(t) \quad (i = T, T-1, \dots). \end{aligned} \right\} \quad (16)$$

Using that for each S-box in DES, all four output bits functionally depend on all six input bits, we obtain

$$X_i(t) = \bigcup_{j \in V_i(t)} V_j, \quad X'_i(t) = \bigcup_{j \in V'_i(t)} V_j. \quad (17)$$

Hence the integers j in $\{1, \dots, 64\}$ such that at least one of the bits d'_j, d''_j (cf. (14)) depends on k_i belong to the set

$$Q(t) = X_M(t) \cup X'_M(t) \cup \left\{ j > 32: j - 32 \in X_{M+1}(t) \cup X'_{M+1}(t) \right\}. \quad (18)$$

It is very easy to compute the sets $Q(t)$, using the recurrence relations (16). For each subset I of $\{1, \dots, 56\}$, the set J of integers in $\{1, \dots, 64\}$ not belonging to any of the sets $Q(t)$ with $t \in I$ has the property that for $j \in J$, both d'_j, d''_j , are functionally independent of the k_i with $i \in I$. The examples for the sets I, J in the table below have been obtained by computing for each j the set of t such that $j \notin Q(t)$. $N = T - R + 1$ denotes the number of consecutive rounds.

R	M	T	N	J	I	$\#I$
1	2	4	4	9,10,11,12	1,3,4,10,14,15,18,25,28,32, 35,38,41,42,44,48,49,52,56	19
1	2	4	4	41,42,43,44	5,9,13,19,20,23,24,26,27,30, 33,36,37,39,43,44,47,51,55	19
1	2	5	5	41,42,43,44	5,20,26,27,30,37,43,44,51	9
1	3	6	6	5,6,7,8	7,28	2
1	3	6	6	17,18,19,20	36,45	2
1	4	7	7	-	-	-
2	5	8	7	5,6,7,8,13,14,15,16	21	1

table 2

In the theorem below we state that non-empty sets I, J of the same type as in table 2 can not be found for blockciphers consisting of 8 or more consecutive rounds of DES.

Theorem 1. Suppose that R, T are integers with $R \geq 1, T \leq 16, T \geq R + 7$. Then for every integer M with $R \leq M \leq T$ and for each t in $\{1, \dots, 56\}$, $Q(t) = \{1, \dots, 64\}$.

Proof: Let t, M be integers with $1 \leq t \leq 56, R \leq M \leq T$. The key scheduling of DES has the property that each integer in $\{1, \dots, 56\}$ is chosen by at least one of the mappings L_i, L_{i+1} for $i = \{1, \dots, 15\}$. This can be verified by using the fact that the only integers in $\{1, \dots, 56\}$ not chosen by PC2 are 9,18,22,25,35,38,43,54. Hence if there is an integer not chosen by L_i and L_j then $r(i) - r(j)$ must be equal to the difference (mod 28) of two of the integers 9,18,22,25 or of two of the integers 35,38,43,54. But $r(i+1) - r(i)$ is either equal to 1 or 2 for $i = 1, \dots, 16$. From the recurrence relations (16) we infer that the sets $V_{R+2}(t), V'_{T-2}(t)$ are non-empty. It is

easy to check from table 1, that \mathcal{F}^2 (\mathcal{F} iterated twice) maps each non-empty subset of $\{1, \dots, 8\}$ onto $\{1, \dots, 8\}$. Again by (16), we infer that $V_{R+4}(t) = V_{R+5}(t) = \dots = \{1, \dots, 8\}$, $V'_{T-4} = V'_{T-5} = \dots = \{1, \dots, 8\}$. We have either $M \geq R+4$, or $M+1 \leq T-4$, or $M+1 = R+4$ and $M = T-4$. In these three cases, we have $X_M(t) \cup X'_{M+1}(t) = X_{M+1}(t) \cup X'_{M+1}(t) = \{1, \dots, 32\}$. This proves Theorem 1. \square

Remark. By a similar method as in the proof of Theorem 1, one can show that in case $T = R+6$, $Q(t)$ can only be a proper subset of $\{1, \dots, 64\}$ if t is not chosen by both L_R and L_T . From the shift-pattern one recovers that $r(R+6) - r(R)$ is equal to either 11 or 12 for $R = 1, 2, \dots, 10$. If PC2 is made in such a way that no difference of the integers not chosen by PC2 is congruent to 1, 2, 11 or 12 (mod 28) then in Theorem 1 we can replace $R+7$ by $R+6$. We do not know if, by a proper choice of PC2, $R+7$ can be replaced by $R+5$.

4. SEQUENCES OF FACTOR SPACES IN DES

In this section we shall investigate the sequences of factor spaces in blockciphers consisting of a reduced number of rounds of DES. We shall use the same notation as in the previous sections. In particular, the blockciphers $F_i: \mathbb{F}_2^{64} \times \mathbb{F}_2^{56} \rightarrow \mathbb{F}_2^{64}$ are defined by $F_i(\mathbf{q}_0, \mathbf{q}_1, \mathbf{k}) = (\mathbf{q}_1, \mathbf{q}_0 + S(EP\mathbf{q}_1 + L_i\mathbf{k}))$ for $(\mathbf{q}_0, \mathbf{q}_1) \in \mathbb{F}_2^{64}$ and $\mathbf{k} \in \mathbb{F}_2^{56}$, and $DES_{R,T} = F_T F_{T-1} \dots F_R$. We shall implicitly assume that the sequences of factor spaces we will consider are all with respect to F_R, \dots, F_T . Our aim is to investigate if $DES_{R,T}$ has sequences of *useful* factor spaces. (A sequence of factor spaces $(\mathcal{V}_{R-1}, \mathcal{V}_R, \dots, \mathcal{V}_T; \mathcal{W})$ is called useful if $\mathcal{W} \neq \{0\}$ and $\mathcal{V}_T \neq \mathbb{F}_2^{64}$).

Example 1. Let $t \in \{1, \dots, 56\}$ and let $X_{R-1}(t), X_R(t), \dots$ be the sets recursively defined by (15). Let \mathcal{V}_i be the spaces of vectors of which the bits with indices outside

$$Y_i(t) = X_i(t) \cup \left\{ j > 32 : j - 32 \in X_{i+1}(t) \right\}$$

are 0 and let \mathcal{W} be the space generated by the vector in \mathbb{F}_2^{56} of which only the t -th bit is equal to 1. Then $(\mathcal{V}_{R-1}, \mathcal{V}_R, \dots, \mathcal{V}_T; \mathcal{W})$ is a sequence of factor spaces for $DES_{R,T}$ and this sequence is useful if and only if $Y_T(t)$ is properly contained in $\{1, \dots, 64\}$.

Example 2. Reeds and Manferdelli [84] introduced the notion of a "per round linear factor" for DES. A per round linear factor is a linear mapping A on \mathbb{F}_2^{48} for which there exist a mapping \tilde{S} with $AEPS = \tilde{S}A$. If there exists a per round linear factor A which is neither invertible nor has the property that AE maps each vector to $\mathbf{0}$ then one can prove that $(\mathcal{V}, \dots, \mathcal{V}_T; \mathcal{W})$ is a useful sequence of factor spaces for $DES_{R,T}$, where

$$\begin{aligned} \mathcal{V} &= \{(\mathbf{v}_0, \mathbf{v}_1) \in \mathbb{F}_2^{64} : AE\mathbf{v}_0 = AE\mathbf{v}_1 = \mathbf{0}\}, \\ \mathcal{W} &= \{\mathbf{k} \in \mathbb{F}_2^{56} : L_R\mathbf{k} = L_{R+1}\mathbf{k} = \dots = L_T\mathbf{k} = \mathbf{0}\}. \end{aligned}$$

If \mathbb{V}, \mathbb{U} are subspaces of $\mathbb{F}_2^{64}, \mathbb{F}_2^{48}$ respectively, then the spaces $S^P(\mathbb{V}), S^K(\mathbb{U})$ are defined by

$$S^P(\mathbb{V}) = \{(\mathbf{v}_1, \mathbf{v}_0 + S(EP\mathbf{v}_1 + \mathbf{c}) + S(\mathbf{c})) : (\mathbf{v}_0, \mathbf{v}_1) \in \mathbb{V}, \mathbf{c} \in \mathbb{F}_2^{48}\},$$

$$S^K(\mathbb{U}) = \{(\mathbf{0}, S(\mathbf{u} + \mathbf{c}) + S(\mathbf{c})) : \mathbf{u} \in \mathbb{U}, \mathbf{c} \in \mathbb{F}_2^{48}\}.$$

Lemma 2. Let $\mathbb{V}_{R-1}, \mathbb{V}_R, \dots, \mathbb{V}_T \subseteq \mathbb{F}_2^{64}, \mathbb{U} \subseteq \mathbb{F}_2^{56}$ be vector spaces. Then the following statements are equivalent:

- (i) $(\mathbb{V}_{R-1}, \mathbb{V}_R, \dots, \mathbb{V}_T; \mathbb{U})$ is a sequence of linear factors for $DES_{R,T}$;
- (ii) $S^P(\mathbb{V}_{i-1}) + S^K(L_i(\mathbb{U})) \subseteq \mathbb{V}_i$ for $i = R, R+1, \dots, T$.

Proof: In view of Lemma 1, it suffices to show that for $i = R, R+1, \dots, T$,

$$S^P(\mathbb{V}_{i-1}) + S^K(L_i(\mathbb{U})) = \{F_i(\mathbf{q}_0 + \mathbf{v}_0, \mathbf{q}_1 + \mathbf{v}_1, \mathbf{k} + \mathbf{w}) + F_i(\mathbf{q}_0, \mathbf{q}_1, \mathbf{k}) : (\mathbf{q}_0, \mathbf{q}_1) \in \mathbb{F}_2^{64}, \mathbf{k} \in \mathbb{F}_2^{56}, (\mathbf{v}_0, \mathbf{v}_1) \in \mathbb{V}, \mathbf{w} \in \mathbb{U}\}. \quad (19)$$

Denote the right-hand side of (19) by \mathbb{W}_i . Let $i \in \{R, \dots, T\}$. It is easy to check that for $(\mathbf{v}_0, \mathbf{v}_1) \in \mathbb{V}, \mathbf{c} \in \mathbb{F}_2^{48}$,

$$(\mathbf{v}_1, \mathbf{v}_0 + S(EP\mathbf{v}_1 + \mathbf{c}) + S(\mathbf{c})) = F_i(\mathbf{v}_0, \mathbf{v}_1, \mathbf{k}) + F_i(\mathbf{0}, \mathbf{0}, \mathbf{k}),$$

where $\mathbf{k} \in \mathbb{F}_2^{56}$ is chosen such that $L_i\mathbf{k} = \mathbf{c}$. This shows that

$$S^P(\mathbb{V}_{i-1}) \subseteq \mathbb{W}_i. \quad (20)$$

It is also easy to verify that for $\mathbf{w} \in \mathbb{U}, \mathbf{c} \in \mathbb{F}_2^{48}$,

$$(\mathbf{0}, S(L_i\mathbf{w} + \mathbf{c}) + S(\mathbf{c})) = F_i(\mathbf{0}, \mathbf{0}, \mathbf{k} + \mathbf{w}) + F_i(\mathbf{0}, \mathbf{0}, \mathbf{k}),$$

where again $\mathbf{k} \in \mathbb{F}_2^{56}$ is chosen such that $L_i\mathbf{k} = \mathbf{c}$. Hence

$$S^K(L_i(\mathbb{U})) \subseteq \mathbb{W}_i. \quad (21)$$

On the other hand, for $(\mathbf{q}_0, \mathbf{q}_1) \in \mathbb{F}_2^{64}, \mathbf{k} \in \mathbb{F}_2^{56}, (\mathbf{v}_0, \mathbf{v}_1) \in \mathbb{V}_{i-1}, \mathbf{w} \in \mathbb{U}$,

$$\begin{aligned} & F_i(\mathbf{q}_0 + \mathbf{v}_0, \mathbf{q}_1 + \mathbf{v}_1, \mathbf{k} + \mathbf{w}) + F_i(\mathbf{q}_0, \mathbf{q}_1, \mathbf{k}) \\ &= (\mathbf{v}_1, \mathbf{v}_0 + S(EP\mathbf{v}_1 + L_i\mathbf{w} + EP\mathbf{q}_1 + L_i\mathbf{k}) + S(EP\mathbf{q}_1 + L_i\mathbf{k})) \\ &= (\mathbf{v}_1, \mathbf{v}_0 + S(EP\mathbf{v}_1 + \mathbf{c}_1) + S(\mathbf{c}_1)) + (\mathbf{0}, S(L_i\mathbf{w} + \mathbf{c}_2) + S(\mathbf{c}_2)), \end{aligned}$$

where

$$\mathbf{c}_1 = L_i\mathbf{w} + EP\mathbf{q}_1 + L_i\mathbf{k}, \quad \mathbf{c}_2 = EP\mathbf{q}_1 + L_i\mathbf{k}.$$

Hence

$$\mathfrak{W}_i \subseteq S^P(\mathfrak{V}_{i-1}) + S^K(L_i(\mathfrak{W})). \quad (22)$$

A combination of (20),(21),(22) yields (19). \square

In the statement of the next lemma, we use the following notation. Define the linear mappings $\rho_i: \mathbb{F}_2^{48} \rightarrow \mathbb{F}_2^6$, $\rho_i^*: \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^{32}$, $\rho_i^{**}: \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^{64}$, $U: \mathbb{F}_2^{64} \rightarrow \mathbb{F}_2^{48}$ as follows:

$$\begin{aligned} \rho_i(\mathbf{x}) &= \mathbf{x}_i \text{ for } \mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_8) \text{ with } \mathbf{x}_1, \dots, \mathbf{x}_8 \in \mathbb{F}_2^6; \\ \rho_i^*(\mathbf{y}) &= (\mathbf{0}, \dots, \mathbf{0}, \mathbf{y}, \mathbf{0}, \dots, \mathbf{0}) \text{ (with } \mathbf{y} \text{ on the } i\text{-th place)}; \\ \rho_i^{**}(\mathbf{y}) &= (\mathbf{0}, \rho_i^*(\mathbf{y})); \\ U(\mathbf{x}, \mathbf{y}) &= EP\mathbf{y}. \end{aligned}$$

Finally, for any subspace \mathfrak{U} of \mathbb{F}_2^6 , we define the spaces $T_j(\mathfrak{U}), T'_j(\mathfrak{U})$ ($j = 1, \dots, 8$) by

$$\begin{aligned} T_j(\mathfrak{U}) &= [S_j(\mathbf{u} + \mathbf{c}) + S_j(\mathbf{c}) : \mathbf{u} \in \mathfrak{U}, \mathbf{c} \in \mathbb{F}_2^6], \\ T'_j(\mathfrak{U}) &= [S_j(\mathbf{u} + \mathbf{c}) + S_j(\mathbf{c}) + S_j(\mathbf{u}) + S_j(\mathbf{0}) : \mathbf{u} \in \mathfrak{U}, \mathbf{c} \in \mathbb{F}_2^6]. \end{aligned}$$

Lemma 3. Let $\mathfrak{V} = [\mathbf{v}_{0j}, \mathbf{v}_{1j} : j = 1, \dots, p] \subseteq \mathbb{F}_2^{64}$, $\mathfrak{W} = [\mathbf{w}_j : j = 1, \dots, q] \subseteq \mathbb{F}_2^{48}$ be vector spaces. Then

$$S^P(\mathfrak{V}) = [\mathbf{v}_{1j}, \mathbf{v}_{0j} + S(EP\mathbf{v}_{1j}) + S(\mathbf{0}) : j = 1, \dots, p] + \sum_{k=1}^8 \rho_k^{**} T'_k\{\rho_k U(\mathfrak{V})\} \quad (23)$$

and

$$S^K(\mathfrak{W}) = [(\mathbf{0}, S(\mathbf{w}_j) + S(\mathbf{0})) : j = 1, \dots, q] + \sum_{k=1}^8 \rho_k^{**} T'_k\{\rho_k(\mathfrak{W})\}. \quad (24)$$

Proof: We shall only prove (23); (24) can be proved in a similar way. For convenience, we introduce the following notation:

$$\begin{aligned} \mathbf{s}(\mathbf{v}_0, \mathbf{v}_1, \mathbf{c}) &= (\mathbf{v}_1, \mathbf{v}_0 + S(EP\mathbf{v}_1 + \mathbf{c}) + S(\mathbf{c})) \text{ for } (\mathbf{v}_0, \mathbf{v}_1) \in \mathbb{F}_2^{64}, \mathbf{c} \in \mathbb{F}_2^{48}; \\ \mathbf{t}'_j(\mathbf{u}, \mathbf{c}) &= S_j(\mathbf{u} + \mathbf{c}) + S_j(\mathbf{c}) + S_j(\mathbf{u}) + S_j(\mathbf{0}) \text{ for } \mathbf{u}, \mathbf{c} \in \mathbb{F}_2^6. \end{aligned}$$

First of all we remark, that for subspaces $\mathfrak{V}_1, \mathfrak{V}_2$ of \mathbb{F}_2^{64} ,

$$S^P(\mathfrak{V}_1 + \mathfrak{V}_2) = S^P(\mathfrak{V}_1) + S^P(\mathfrak{V}_2) \quad (25)$$

and that for subspaces $\mathcal{Q}_{\mathcal{L}_1}, \mathcal{Q}_{\mathcal{L}_2}$ of \mathbb{F}_2^6 ,

$$T'_j(\mathcal{Q}_{\mathcal{L}_1} + \mathcal{Q}_{\mathcal{L}_2}) = T'_j(\mathcal{Q}_{\mathcal{L}_1}) + T'_j(\mathcal{Q}_{\mathcal{L}_2}) \text{ for } j = 1, \dots, 8. \quad (26)$$

(25) follows easily from the identity

$$\begin{aligned} s(\mathbf{v}'_0 + \mathbf{v}''_0, \mathbf{v}'_1 + \mathbf{v}''_1, \mathbf{c}) &= s(\mathbf{v}'_0, \mathbf{v}'_1, \mathbf{c} + EP\mathbf{v}''_1) + s(\mathbf{v}''_0, \mathbf{v}''_1, \mathbf{c}) \\ &\text{for } (\mathbf{v}'_0, \mathbf{v}''_1), (\mathbf{v}''_0, \mathbf{v}''_1) \in \mathbb{F}_2^{64}, \mathbf{c} \in \mathbb{F}_2^{48}, \end{aligned}$$

while (26) is an easy consequence of the identity

$$\begin{aligned} t'_j(\mathbf{u}' + \mathbf{u}'', \mathbf{c}) &= t'_j(\mathbf{u}', \mathbf{u}'' + \mathbf{c}) + t'_j(\mathbf{u}'', \mathbf{c}) + t'_j(\mathbf{u}', \mathbf{u}'') \\ &\text{for } \mathbf{u}', \mathbf{u}'', \mathbf{c} \in \mathbb{F}_2^6, j = 1, \dots, 8. \end{aligned}$$

In view of (25), (26), it suffices to prove (23) for $p = 1$. Let $\mathcal{V} = \{(\mathbf{v}_0, \mathbf{v}_1)\}$ and put

$$\begin{aligned} \tilde{\mathcal{V}} &= [(\mathbf{v}_1, \mathbf{v}_0 + S(EP\mathbf{v}_1) + S(\mathbf{0})) = [s(\mathbf{v}_0, \mathbf{v}_1, \mathbf{0})], \\ \tilde{\mathcal{Q}}_j &= \rho_j^{\bullet\bullet} T'_j \{ \rho_j U(\{v_1\}) \} = [\rho_j^{\bullet\bullet} t'_j(\rho_j EP\mathbf{v}_1, \mathbf{c}) : \mathbf{c} \in \mathbb{F}_2^6] \text{ for } j = 1, \dots, 8. \end{aligned}$$

From the identity

$$s(\mathbf{v}_0, \mathbf{v}_1, \mathbf{c}) = s(\mathbf{v}_0, \mathbf{v}_1, \mathbf{0}) + \sum_{k=1}^8 \rho_k^{\bullet\bullet} t'_k(\rho_k EP\mathbf{v}_1, \rho_k \mathbf{c}) \text{ for } \mathbf{c} \in \mathbb{F}_2^{48}, \quad (27)$$

it follows easily that

$$S^P(\mathcal{V}) \subseteq \tilde{\mathcal{V}} + \sum_{k=1}^8 \tilde{\mathcal{Q}}_k. \quad (28)$$

On the other hand we have

$$\tilde{\mathcal{V}} \subseteq S^P(\mathcal{V}). \quad (29)$$

Let $\mathbf{d} \in \mathbb{F}_2^6$ and choose $\mathbf{c} \in \mathbb{F}_2^{48}$ such that $\rho_j(\mathbf{c}) = \mathbf{d}, \rho_k(\mathbf{c}) = \mathbf{0}$ for $k \neq j$. Then (27) implies that

$$\rho_j^{\bullet\bullet} t'_j(\rho_k EP\mathbf{v}_1, \mathbf{d}) = s(\mathbf{v}_0, \mathbf{v}_1, \mathbf{c}) + s(\mathbf{v}_0, \mathbf{v}_1, \mathbf{0}).$$

Hence

$$\tilde{\mathcal{Q}}_j \subseteq S^P(\mathcal{V}) \text{ for } j = 1, \dots, 8. \quad (30)$$

Now (23) follows at once from (28), (29), (30). \square

Lemma 3 shows that for each subspace \mathcal{V} of \mathbb{F}_2^{64} (\mathcal{W} of \mathbb{F}_2^{56}) the space $S^P(\mathcal{V}), (S^K(\mathcal{W}))$ can be expressed as the sum of a space generated by a set of vectors of which the cardinality is not larger than the dimension of $\mathcal{V}(\mathcal{W})$ and spaces which can be described completely in terms of the S-boxes. Thus Lemma 3 provides us a rather efficient method which checks if a given sequence of spaces $(\mathcal{V}_{R-1}, \mathcal{V}_R, \dots, \mathcal{V}_T, \mathcal{W})$ is a sequence of factor spaces for $DES_{R,T}$. From the arguments used in the proof of Lemma 3 it is clear, that this method can be applied also to a general class of block ciphers which can be described in the same way as DES, with arbitrary S-boxes (which can be different in each round), an arbitrary linear mapping instead of EP (where it is allowed that in different rounds different linear mappings are chosen) and arbitrary surjective linear mappings instead of the L_i .

We shall now give explicit expressions for the spaces $S^P(\mathcal{V}), S^K(\mathcal{W})$. For this purpose we have only to compute the spaces $T'(\mathcal{U})$ with $\mathcal{U} \subseteq \mathbb{F}_2^6$.

Lemma 4. For all g in $\{1, \dots, 8\}$ and all subspaces \mathcal{U} of \mathbb{F}_2^6 with $\mathcal{U} \neq \{0\}$, we have $T_g(\mathcal{U}) = T'_g(\mathcal{U}) = \mathbb{F}_2^4$, with the following exceptions:

$$\begin{aligned} T_4(\{000001\}) &= \{1100, 0011, 1010\}, T'_4(\{000001\}) = \{1100, 0011\}; \\ T_4(\{101110\}) &= T'_4(\{101110\}) = \{1010, 0101\}; \\ T_4(\{101111\}) &= T'_4(\{101111\}) = \{1001, 0110\}; \\ T_4(\{000001, 101110\}) &= T'_4(\{000001, 101110\}) = \{1100, 0011, 1010\}. \end{aligned}$$

Proof: This can be verified by straightforward computation. \square

In the theorem below, the sets $S^P(\mathcal{V}), S^K(\mathcal{W})$, with \mathcal{V}, \mathcal{W} being subspaces of $\mathbb{F}_2^{64}, \mathbb{F}_2^{48}$ respectively, are defined by

$$\begin{aligned} S^P(\mathcal{V}) &= \{g: 1 \leq g \leq 8, \rho_g U(\mathcal{V}) \neq \{0\}\}, \\ S^K(\mathcal{W}) &= \{g: 1 \leq g \leq 8, \rho_g(\mathcal{W}) \neq \{0\}\}. \end{aligned}$$

Theorem 2. Let \mathcal{V}, \mathcal{W} be subspaces of $\mathbb{F}_2^{64}, \mathbb{F}_2^{48}$ respectively. Then

$$(i) \quad S^P(\mathcal{V}) = \tilde{\mathcal{V}} + \sum_{g \in S^P(\mathcal{V})} \tilde{\mathcal{V}}_g,$$

where

$$\tilde{\mathcal{V}} = \{(\mathbf{v}_1, \mathbf{v}_0): (\mathbf{v}_0, \mathbf{v}_1) \in \mathcal{V} \mid \text{if } \rho_4 U(\mathcal{V}) \neq \{000001\}\},$$

$$\tilde{\mathcal{V}} = \left\{ (\mathbf{v}_1, \mathbf{v}_0 + \alpha \rho_4^*(1010)): (\mathbf{v}_0, \mathbf{v}_1) \in \mathcal{V}, \begin{cases} \alpha = 0 & \text{if } \rho_4 EP \mathbf{v}_1 = 000000 \\ \alpha = 1 & \text{if } \rho_4 EP \mathbf{v}_1 = 000001 \end{cases} \right\}$$

if $\rho_4 U(\mathcal{V}) = [000001]$;

and where

$$\mathcal{V}_g = \rho_g^{**} \mathbb{F}_2^4 \text{ for } g \in \mathbb{S}^P(\mathcal{V})$$

with the following exceptions if $4 \in \mathbb{S}^P(\mathcal{V})$:

$$\mathcal{V}_4 = \rho_4^{**} ([1100, 0011]) \text{ if } \rho_4 U(\mathcal{V}) = [000001] ;$$

$$\mathcal{V}_4 = \rho_4^{**} ([1010, 0101]) \text{ if } \rho_4 U(\mathcal{V}) = [101110] ;$$

$$\mathcal{V}_4 = \rho_4^{**} ([1001, 0110]) \text{ if } \rho_4 U(\mathcal{V}) = [101111] ;$$

$$\mathcal{V}_4 = \rho_4^{**} ([1100, 0011, 1010]) \text{ if } \rho_4 U(\mathcal{V}) = [000001, 101110] .$$

$$(ii) \quad S^K(\mathcal{W}) = \sum_{g \in \mathbb{S}^K(\mathcal{W})} \mathcal{W}_g ,$$

where $\mathcal{W}_g = \rho_g^{**} \mathbb{F}_2^4$ for $g \in \mathbb{S}^K(\mathcal{W})$ with the following exceptions if $4 \in \mathbb{S}^K(\mathcal{W})$:

$$\mathcal{W}_4 = \rho_4^{**} ([1010, 0101]) \text{ if } \rho_4(\mathcal{W}) = [101110] ;$$

$$\mathcal{W}_4 = \rho_4^{**} ([1001, 0110]) \text{ if } \rho_4(\mathcal{W}) = [101111] ;$$

$$\mathcal{W}_4 = \rho_4^{**} ([1100, 0011, 1010]) \text{ if } \rho_4(\mathcal{W}) = [000001] \text{ or } [000001, 101110] .$$

Proof: The proofs of (i),(ii) can be derived easily from Lemmas 3,4. We shall only give a rough sketch of the proof of (i). By Lemma 4,

$$\rho_g^{**} \left[S_g(\rho_g \mathbf{v}_1) + S_g(\mathbf{0}) \right] \in \mathcal{V}_g \text{ for } g \in \mathbb{S}^P(\mathcal{V})$$

except when $g=4$, $\rho_4 EP \mathbf{v}_1 = 000001$. This proves that for $(\mathbf{v}_0, \mathbf{v}_1) \in \mathcal{V}$, there is a vector

$$\mathbf{u} \in \sum_{g \in \mathbb{S}^P(\mathcal{V})} \mathcal{V}_g, \text{ with}$$

$$(\mathbf{v}_1, \mathbf{v}_0 + S(EP \mathbf{v}_1) + S(\mathbf{0})) = (\mathbf{v}_1, \mathbf{v}_0 + \mathbf{u}) \text{ if } \rho_4 EP \mathbf{v}_1 \neq 000001 ,$$

$$\begin{aligned} (\mathbf{v}_1, \mathbf{v}_0 + S(EP \mathbf{v}_1) + S(\mathbf{0})) &= \\ &= \left[\mathbf{v}_1, \mathbf{v}_0 + \rho_4^{**} \{ S_4(\rho_4 EP \mathbf{v}_1) + S_4(\mathbf{0}) \} \right] + \mathbf{u} \\ &= \left[\mathbf{v}_1, \mathbf{v}_0 + \rho_4^{**} (1010) \right] + \mathbf{u} \\ &\text{if } \rho_4 EP \mathbf{v}_1 = 000001. \end{aligned}$$

These facts immediately prove (i). \square

We shall now prove that blockciphers consisting of eight or more consecutive rounds of DES are resistant against a meet-in-the-middle attack using sequences of factor spaces. To this end we shall need the following lemma.

Lemma 5. *Let $T \geq R + 3$. If $(\mathcal{V}_{R-1}, \mathcal{V}_R, \dots, \mathcal{V}_T; \mathcal{U})$ is a sequence of factor spaces for $DES_{R,T}$ with $\mathcal{U} \neq \{0\}$, then*

- (i) $\mathcal{V}_{R+3} \supseteq \{(0, y) : y \in \mathbb{F}_2^{32}\}$,
- (ii) $\mathcal{V}_{R+i} = \mathbb{F}_2^{64}$ for $i \geq 4$.

Proof: (ii) is an immediate consequence of (i) and Lemmas 2,4. We shall now prove (i). Since all elements of $\{1, \dots, 56\}$ are chosen by at least one of the mappings L_R, L_{R+1} , at least one of the spaces $S^K(L_R(\mathcal{U})), S^K(L_{R+1}(\mathcal{U}))$ is $\neq \{0\}$. By Lemma 2 and Theorem 2,

$$\mathcal{V}_{R+1} \supseteq \sum_{g \in \mathcal{S}} \mathcal{V}_g^1$$

for some non-empty subset \mathcal{S} of $\{1, \dots, 8\}$, where $\mathcal{V}_g^1 = \rho_g^{**}(\mathbb{F}_2^4)$ if $g \neq 4$ and $\mathcal{V}_4^1 = \rho_4^{**}(\mathcal{U}^1)$ with \mathcal{U}^1 being a subspace of \mathbb{F}_2^4 with $\mathcal{U}^1 \neq \{0\}$ in case that $4 \in \mathcal{S}$. The space \mathcal{U}^1 has the property that for each j in $\{1, \dots, 4\}$ there is a vector $x_1 x_2 x_3 x_4$ in \mathcal{U}^1 with $x_j \neq 0$. *EP* sends the indices of the output bits of S-box S_j (i.e the elements of $\{4j-3, \dots, 4j\}$) to the indices of the input bits of 6 different S-boxes, namely the S-boxes S_k with $k \in \mathcal{F}(\{j\})$, where \mathcal{F} is the function defined in table 1. Together with Lemma 2 and Theorem 2 these facts imply that

$$\mathcal{V}_{R+2} \supseteq \sum_{g \in \mathcal{F}(\mathcal{S})} \mathcal{V}_g^2,$$

where $\mathcal{V}_g^2 = \rho_g^{**}(\mathbb{F}_2^4)$ for $g \neq 4$ and when $4 \in \mathcal{F}(\mathcal{S})$, $\mathcal{V}_4^2 = \rho_4^{**}(\mathcal{U}^2)$ for some subspace \mathcal{U}^2 of \mathbb{F}_2^4 with $\mathcal{U}^2 \neq \{0\}$. We remark that

$$S^P(\rho_g^{**}(\mathbb{F}_2^4)) \supseteq \rho_4^{**}(\mathbb{F}_2^4) \text{ for } g = 2, 3, 5, 7, 8, \quad (31)$$

hence $\mathcal{U}^2 = \mathbb{F}_2^4$ if one of the numbers 2,3,5,7,8 belongs to \mathcal{S} . Since $\mathcal{F}(\mathcal{S})$ has cardinality at least 6, at least one of the numbers 2,3,5,7,8 belongs to $\mathcal{F}(\mathcal{S})$. By repeating the argument from above, and using that \mathcal{F}^2 maps each non-empty subset of $\{1, \dots, 8\}$ onto $\{1, \dots, 8\}$, we obtain

$$\mathcal{V}_{R+3} \supseteq \sum_{g \in \mathcal{F}^2(\mathcal{S})} \rho_g^{**}(\mathbb{F}_2^4) = \{(0, y) : y \in \mathbb{F}_2^{32}\}.$$

This completes the proof of Lemma 5. \square

Lemma 5 includes the result of Meyer mentioned in §1. Another consequence of Lemma 5 is that the only per round linear factors of DES are the linear mappings $A : \mathbb{F}_2^{48} \rightarrow \mathbb{F}_2^{48}$ for which either A is invertible or AE maps each vector of \mathbb{F}_2^{32} onto 0. (cf. example 2 at the beginning of

this section.) This fact was already proved by Reeds and Manferdelli [84].

We shall now prove our final result.

Theorem 3. *Let R, M, T be integers with $1 \leq R \leq M \leq T \leq 16$ and $T \geq R + 7$. Let $(\mathcal{V}_{R-1}, \mathcal{V}_R, \dots, \mathcal{V}_M; \mathcal{W})$, $(\mathcal{V}'_T, \mathcal{V}'_{T-1}, \dots, \mathcal{V}'_M; \mathcal{W})$ be sequences of factor spaces for $DES_{R,M}, DES_{M+1,T}^{-1}$, respectively, such that $\mathcal{W} \neq \{0\}$ and $\mathcal{V}_M = \mathcal{V}'_M$. Then*

$$\mathcal{V}_M = \mathcal{V}'_M = \mathbb{F}_2^{64}.$$

Proof: In the proof we shall use that the inverse of a round of DES (i.e. one of the blockciphers F_i) is equal to the round itself, except that the left half and the right half of both plaintext and ciphertext must be interchanged.

We distinguish three cases: (i) $M \geq R + 4$; (ii) $M \leq T - 5$; (iii) $M = R + 3 = T - 4$. In case (i) we have $\mathcal{V}_M = \mathbb{F}_2^{64}$, by Lemma 5, (ii). In case (ii) we can prove, completely similar to Lemma 5, (ii), that $\mathcal{V}_M = \mathbb{F}_2^{64}$, using that all elements of $\{1, \dots, 56\}$ are chosen by at least one of the mappings L_T, L_{T-1} . In case (iii) we have firstly, by Lemma 5, (i), $\mathcal{V}_M = \mathcal{V}_{R+3} \supseteq \{(0, y) : y \in \mathbb{F}_2^{32}\}$. By an argument completely similar to the proof of Lemma 5 (i), one has $\mathcal{V} = \mathcal{V}_{T-4} \supseteq \{(x, 0) : x \in \mathbb{F}_2^{32}\}$. This completes the proof of Theorem 3. \square

Remark. By changing $PC2$ in the way described at the end of §3, it is possible to replace the condition $T \geq R + 7$ by $T \geq R + 6$ in Theorem 3. This can be proved in a similar way as Theorem 3.

CONCLUDING REMARKS

Linear structures allowing known-plaintext attacks on blockciphers have been investigated, particularly those consisting of a reduced number of consecutive rounds of DES. The first structures we looked for were “bits in the middle” independent of key bits. Such independencies were found only in blockciphers comprising less than eight rounds of DES. We discovered that $PC2$ was not optimal in the sense that by a change of $PC2$ blockciphers of seven instead of eight consecutive rounds of DES would have no “bits in the middle” independent of key bits. The existence of such independencies in blockciphers for such numbers of rounds depends only on the structure of the tables E , P , and $PC2$; these independencies would hold for any S-boxes. More general linear structures were also considered, namely sequences of linear factors. The existence of these factors depends not only on the structure of E , P , and $PC2$, but also on the structure of the S-boxes. In spite of some linear structure in S-box 4, we were able to show that blockciphers consisting of eight or more consecutive rounds of DES do not have sequences of linear factors with respect to these rounds that can reduce the search time for the key in a meet-in-the-middle attack.

A natural extension of the attacks described in this paper would seek changes in the tables defining the S-boxes that yield S-boxes with linear factors cooperating to give useful sequences of linear factors. (One might even change the S-boxes differently in different rounds.) Any sequence of linear factors for the cipher with the modified S-boxes is then a sequence of "near" linear factors for the original cipher. (As has been pointed out in §2, such attacks generalize several ideas in [Hellman et al 76] and [Desmedt, Quisquater and Davio 84].) In this way one might obtain sequences of near linear factors that allow cryptanalysis of blockciphers consisting of eight or more rounds of DES.

REFERENCES

- (1) National Bureau of Standards, "Data Encryption Standard", U.S. Department of Commerce, FIPS pub. 46 (January 1977).
- (2) Davio, M., Desmedt, Y., Fosséprez, M., Govaerts, R., Hulsbosch, J., Neutjens, P., Piret, P., Quisquater, J.J., Vandewalle, J., Wouters, P., "Analytical characteristics of the DES," in *Advances in Cryptology: Proc. Crypto '83*, D. Chaum, ed., Plenum, New York (1984), pp. 171-202.
- (3) Desmedt, Y., Quisquater, J.J., Davio, M., "Dependence of output on input in DES: Small avalanche characteristics," in *Advances in Cryptology: Proc. Crypto '84*, G.R. Blakley and D. Chaum, eds., *Lecture Notes in Computer Science* 196, Springer-Verlag, Berlin (1985), pp. 359-376.
- (4) Hellman, M., Merkle, R., Schroepfel, R., Washington, L., Diffie, W., Pohlig, S., Schweitzer, P., "Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard," Information Systems Lab. report SEL 76-042, Stanford University (1976).
- (5) Meyer, C.H., "Ciphertext-plaintext and ciphertext-key dependencies vs. number of rounds for the Data Encryption Standard," *AFIPS Conference Proceedings*, 47, (June 1978), pp. 1119-1126.
- (6) Reeds, J.A., Manferdelli, J.L., "DES has no per round linear factors," in *Advances in Cryptology: Proc. Crypto '84*, G.R. Blakley and D. Chaum, eds., *Lecture Notes in Computer Science* 196, Springer-Verlag, Berlin (1985), pp. 377-389.

Is DES a Pure Cipher? (Results of More Cycling Experiments on DES)¹

(Preliminary Abstract)

Burton S. Kaliski Jr., Ronald L. Rivest, and Alan T. Sherman

*MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
December 1985*

Abstract

During summer 1985, we performed eight cycling experiments on the Data Encryption Standard (DES) to see if DES has certain algebraic weaknesses. Using special-purpose hardware, we applied the cycling closure test described in our Eurocrypt 85 paper to determine whether DES is a pure cipher. We also carried out a stronger version of this test. (A cipher is *pure* if, for any keys i, j, k , there exists some key l such that $T_i T_j^{-1} T_k = T_l$, where T_w denotes encryption under key w .) In addition, we followed the orbit of a randomly chosen DES transformation for 2^{36} steps, as well as the orbit of the composition of two of the “weak key” transformations. Except for the weak key experiment, our results are consistent with the hypothesis that DES acts like a set of randomly chosen permutations. In particular, our results show with overwhelming confidence that DES is not pure. The weak key experiment produced a short cycle of about 2^{33} steps, the consequence of hitting a fixed point for each weak key.

Key Words and Phrases

Birthday Paradox, closed cipher, cryptanalysis, cryptography, cryptology, cycle-detection algorithm, Data Encryption Standard (DES), finite permutation group, idempotent cryptosystem, multiple encryption, pure cipher.

¹This research is supported by NSF grant MCS-8006938 and IBM.

1 Introduction

At the Eurocrypt 85 conference, we presented experimental statistical evidence that the set of DES transformations is not closed under functional composition [KRS85].² During May to August 1985, we performed additional experiments to determine if DES has certain other related algebraic weaknesses. In particular, we addressed the open question, “Is DES a pure cipher?”³ In addition, we performed a strengthened version of our closure test and we ran two experiments to investigate the order of DES transformations. Using a combination of software and special-purpose hardware, we carried out eight experiments, covering five different algebraic tests. Although we experimented only with DES, our tests are general in nature and apply to any to finite, deterministic cryptosystem.

None of our experiments involving randomly chosen DES transformations detected any algebraic weaknesses. In particular, our data show with extremely high confidence that DES is not pure. However, one experiment inadvertently discovered fixed points for two of the keys, thereby revealing a previously unpublished additional weakness of the weak keys [Dav82].

This abstract is organized in four sections. Section 1 gives an overview of our experiments and explains the purpose of our tests. Section 2 introduces the notation and terminology used throughout the abstract and summarizes previous cycling studies on DES. Section 2 also briefly reviews the cycling closure test and describes our hardware implementation of it. Section 3 lists concise descriptions of our algebraic tests. Finally, section 4 summarizes our findings and explains the two interesting structural properties that we encountered during our tests. An appendix which describes our detailed experimental results is also included.

1.1 Overview and Motivation

It is important to know if DES is pure for essentially the same reasons that it is important to know if DES is closed. If DES were pure, then Tuchman’s multiple encryption scheme would be equivalent to single encryption, and DES would be vulnerable to a known-plaintext attack that runs in 2^{28} steps on the average [KRS85].⁴ It is possible that DES is pure, but not closed. (Of course, if DES were closed, then DES would also be pure.) Although there is no particular reason to suspect that DES is pure, it is unknown in the open literature if DES has this weakness.

The question “Is DES closed?” is a question about the order of the group generated by DES. A related and more detailed question—which we call the *small subgroup question*—is: “What is the order of the group generated by n given DES transformations?” Any set of DES transformations that generates a small group would suffer the weaknesses of closed ciphers. Specifically, any such set of transformations would be vulnerable to our known-plaintext attack against closed ciphers. In addition, multiple encryption (using either sequential multiple encryption or Tuchman’s scheme) involving only transformations from such a set would be equivalent to single encryption from the set.⁵ Finally, when used in output-feedback mode with feedback width 64 [FIS80], any transformation from such a set would be at greater risk to produce a key stream with short period.

²The Data Encryption Standard (DES) is a federal standard for the cryptographic protection of computer data, adopted in November 1976 by the United States National Bureau of Standards (NBS) [FIPS77, DaP84].

³See section 2.1 for a review of the definition of a pure cipher.

⁴To encrypt a message x under Tuchman’s scheme is to compute $T_i T_j^{-1} T_k(x)$, where the keys i , j , and k are chosen independently [Tuc78, McM82].

⁵To encrypt a message x using sequential multiple encryption is to compute $T_i T_j(x)$, where the keys i and j are chosen independently [McH81].

Two of our tests address the small subgroup question for $n = 1, 2$.

To test DES for purity and other algebraic weaknesses, we examined the orbits of subsets of DES transformations on particular messages. Our method was to compute the orbits of single DES transformations and to apply our cycling closure test to subsets of two or more DES transformations. To carry out the tests we built special-purpose hardware and implemented a variation of the constant-space cycle-detection algorithm described by Sedgewick and Szymanski [SSY82]. We applied our tests both to randomly chosen transformations and to transformations with special properties (e.g. transformations represented by weak keys). The dominant theme of our tests was to determine if DES has algebraic properties different from those expected from a set of randomly selected permutations.

Since there is an overwhelming chance that even two randomly selected permutations will generate either the alternating group or the symmetric group [BoW77,Dix69], we did not expect to detect any pairs of DES transformations that generate small groups.

2 Background

2.1 Definitions and Notation

The Data Encryption Standard (DES) specifies a mapping $T : K \times M \rightarrow M$, where $K = \{0, 1\}^{56}$ is the *key space* and $M = \{0, 1\}^{64}$ is the *message space*. Each key $k \in K$ represents a transformation $T_k = T(k, \cdot)$, which, by the definition of DES, permutes M . DES is *endomorphi*c: its message space and ciphertext space are the same set. It is unknown if DES is *faithful*: does every key represent a distinct permutation?

We shall use the following notations throughout the paper. Let $M = |M| = 2^{64}$ be the *degree* of DES; let $K = |K| = 2^{56}$ be the size of the key space; and let $\mathcal{T} = \bigcup \{T_k : k \in K\}$ be the set of all DES transformations. In addition, for any transformation $T_k \in \mathcal{T}$, let T_k^{-1} denote the inverse of T_k .

Let I be the identity permutation on M , and let \mathcal{A}_M and \mathcal{S}_M be, respectively, the *alternating group* and *symmetric group* on M .⁶ For any permutations g, h we denote the composition of g and h by $gh = g \circ h = g[h(\cdot)]$. For any permutations g_1, g_2, \dots, g_n , let $\langle g_1, g_2, \dots, g_n \rangle$ denote the group generated by g_1, g_2, \dots, g_n . Of course, for any n DES transformations T_1, T_2, \dots, T_n , it is true that $\langle T_1 \rangle \subseteq \langle T_1, T_2 \rangle \subseteq \langle T_1, T_2, \dots, T_n \rangle \subseteq \langle \mathcal{T} \rangle$. Since each round of DES is an even permutation, it is also true that $\langle \mathcal{T} \rangle \subseteq \mathcal{A}_M$.

For any subgroup $G \subseteq \mathcal{S}_M$, for any $x \in M$, the *G-orbit* of x is the set $G\text{-orbit}(x) = \{g(x) : g \in G\}$. For any permutation $g \in \mathcal{S}_M$, may write $g\text{-orbit}(x)$ to denote the $\langle g \rangle$ -orbit of x . If f is any function (not necessarily a permutation) and if $x \in \text{Domain}(f)$, we define the *f-closure* of x to be the set $f\text{-closure}(x) = \{f^i(x) : i \geq 0\}$. For any subgroup $G \subseteq \mathcal{S}_M$, the *order* of G is the number of elements in G . For any $g \in \mathcal{S}_M$, the *order* of g is the order of $\langle g \rangle$.

A cryptosystem is *closed* if and only if its set of encryption transformations is closed under functional composition, i.e. DES is closed if and only if for all keys $i, j \in K$ there exists a key $k \in K$ such that $T_i T_j = T_k$.⁷ Since every finite cancellation semigroup is a group, DES is closed if and only if \mathcal{T} forms a group under functional composition.

⁶See [Car56], [Rot78], or [Wie64] for a review of basic concepts in permutation group theory.

⁷Note that we are using the term *closed cipher* to refer to what Shannon called an *idempotent cipher* [Sha49]. Shannon defined a closed cipher to be any cryptosystem with the property that each cryptographic transformation is surjective.

Shannon's notion of a pure cipher generalizes the idea of closure to non-endomorphic cryptosystems [Sha49]. DES is *pure* if and only if, for every keys $i, j, k \in \mathcal{K}$, there exists a key $l \in \mathcal{K}$ such that $T_i T_j^{-1} T_k = T_l$.⁸ It is easy to see that DES is pure if and only if for every $T_0 \in \mathcal{T}$ the set $T_0^{-1} \mathcal{T}$ is closed. Moreover, $T_0^{-1} \mathcal{T}$ is closed for every $T_0 \in \mathcal{T}$ if and only if $T_0^{-1} \mathcal{T}$ is closed for some $T_0 \in \mathcal{T}$. Every closed cryptosystem is pure, but not every endomorphic pure cryptosystem is closed.

Finally, for any any string $s \in \{0, 1\}^*$, let \bar{s} denote the bitwise complement of s .

2.2 Previous Cycling Studies on DES

To the best of our knowledge, the small subgroup question for two or more DES transformations had not been previously investigated in the open literature. A few researchers have, however, studied the pseudo-random key streams produced by DES in output-feedback mode [FIS80]. Whenever the feedback width is 64 bits, each such key stream describes the orbit of a DES transformation on some initial message. In a series of software experiments, Gait computed the key stream produced by DES in output-feedback mode to at most $10^6 \approx 2^{20}$ places. He found no cycles for nonweak keys [Gai77]. Gait did not state what feedback width he used. Davies and Price [DaP82, DaP82a] and Jueneman [Jue82] studied mathematically the cycle structure of the key stream produced in output-feedback mode, but did not report performing any experiments on DES. Davies and Price did run a series of experiments on random permutations on $\{0, 1\}^8$ [DaP82a]. Finally, in a series of experiments, Hellman and Reyneri investigated the cycle structure of mappings induced by DES on the key space [HeR82]. None of these studies answered the question, "Is DES pure?"

2.3 Review of Cycling Closure Test

The cycling closure test is a statistical test that explores one aspect of the algebraic structure of any finite, deterministic cryptosystem. It works by taking a pseudo-random walk in the message space for a specified number of steps or until a cycle is detected. For each step of the pseudo-random walk, the previous ciphertext is encrypted under a key chosen by a pseudo-random function of the previous ciphertext. Results of the test are asymmetrical: long walks are overwhelming evidence that the set of permutations is not a group; short walks are strong evidence that the set of permutations has a structure different from that expected from a set of randomly chosen permutations [KRS85].

When applied to DES and given an initial message x_0 , the cycling closure test computes the ψ_ρ -closure of x_0 , where the function $\psi_\rho : \mathcal{M} \rightarrow \mathcal{M}$ is defined by $\psi_\rho(x) = T_{\rho(x)}(x)$ whenever $x \in \mathcal{M}$, and $\rho : \mathcal{M} \rightarrow \mathcal{K}$ is a deterministic pseudo-random function. If ρ is "random," then ψ_ρ acts like a random function on the $\langle \mathcal{T} \rangle$ -orbit of x_0 . The expected length of the ψ_ρ -closure computed by the test is about the square root of the length of the $\langle \mathcal{T} \rangle$ -orbit of x_0 .

When applied to a subset $S \subseteq \mathcal{T}$ of two or more DES transformations, the cycling closure test computes the ψ_ρ -closure of x_0 , where $\rho : \mathcal{M} \rightarrow H$ and $H \subseteq \mathcal{K}$ is a set of keys that represents S .

If DES acts like a set of randomly chosen permutations, then we would expect $\langle \mathcal{T} \rangle$ -orbit(x_0) = \mathcal{M} , in which case we would expect $|\psi_\rho$ -closure(x_0)| $\approx \sqrt{M} = 2^{32}$. However, if DES were closed, then $|\langle \mathcal{T} \rangle$ -orbit(x_0) $\leq K$, in which case we would expect $|\psi_\rho$ -closure(x_0)| $\leq \sqrt{K} = 2^{28}$.

⁸Shannon also required each transformation of a pure cipher to be equally likely.

The cycling closure test collects evidence which can be used to compute a measure of our relative degree of belief in the following two competing hypotheses:

- H_G = "DES is a group."
- H_R = "Each DES transformation was chosen independently with uniform probability from the symmetric group on M ."

Let E be the evidence that a trial of the cycling closure test ran for r steps without detecting a cycle. As explained in [KRS85], this evidence can be interpreted by computing the conditional probabilities $p_G = P(E \mid H_G)$ and $p_R = P(E \mid H_R)$, where

$$p_G \approx e^{-r^2/2K} \text{ and } p_R \approx e^{-r^2/2M}. \quad (1)$$

In light of the evidence E , a Bayesian would update her initial odds in favor of H_G over H_R by a factor of p_G/p_R .

2.4 Special-Purpose Hardware

We carried out each experiment using special-purpose hardware which we had originally built to test DES for closure. The main feature of our hardware is that it can compute a sequence of 2^{32} DES encryptions per day, where at each step the previous ciphertext is encrypted under a key that depends on the previous ciphertext. Our hardware consists of a custom wire-wrap board that plugs into an IBM personal computer. The board contains one AMD Z8068 DES chip and a 7.1 MHz finite state controller. By modifying the microcode of the board's finite-state controller, we adapted the board to carry out each of the five algebraic tests. (See [KRS85] for a more detailed description of our special-purpose hardware.⁹)

3 Cycling Experiments on DES

This section briefly describes the four additional cycling tests that we performed on DES. We call these tests the *purity test*, *orbit test*, *small subgroup test*, *closure test*, and *extended message space closure test*. A sixth *reduced message space test* is also described.

3.1 Purity Test

Pick any transformation $T_0 \in \mathcal{T}$ and apply the cycling closure test to the set $T_0^{-1}\mathcal{T}$. (See section 2.3 for a review of the cycling closure test.)

3.2 Orbit Test

Given any key k and any message x_0 , compute $x_i = T_k^i(x_0)$, $i = 1, 2, \dots$ for a specified number of steps or until a cycle is detected.

The period of this sequence is the length of T_k -orbit(x_0). In other words, if we consider the permutation T_k as a product of disjoint cycles, then the period of the sequence is simply the

⁹Schematic diagrams of our hardware will be included in a revised version of this paper, to be available from the authors some time in the future.

length of the cycle that contains x_0 . If this test is run for r steps without detecting a cycle, then r is a lower bound on $\text{order}(T_k)$ and hence on $\text{order}(\langle T \rangle)$.

For a randomly chosen permutation on M , for each $1 \leq l \leq M$, the probability that x_0 lies in a cycle of length exactly l is $1/M$ [Har59, PuW68] ([Knu69], exercise 3.1.12). Hence, the expected cycle-length of the longest cycle of a randomly chosen permutation on n letters is about $0.624n$ [ShL66] (for DES, this is about 2^{63}). For a randomly chosen permutation on M , the chance that we fall into a cycle of length 2^{36} or less is about $2^{-(63-36)} = 2^{-27}$.

Although we do not do so in this preliminary abstract, it is possible to interpret results of the orbit test to obtain statistical lower bounds on the order of the group generated by DES. Such analysis depends on the structure of the group. For example, the orbit test behaves differently on cyclic groups than on symmetric groups. Consequently, it is useful to combine the orbit test with other algebraic tests, including tests for faithfulness, commutativity, solvability at various levels, and nilpotence at various classes.

3.3 Small Subgroup Test

Given two distinct keys $i, j \in K$ and any message x_0 , apply the cycling closure test to the set $\{T_i, T_j\}$ to obtain a statistical lower bound on the length of the $\langle T_i, T_j \rangle$ -orbit of x_0 .

In the orbit and small group tests, it would be interesting to examine both randomly chosen transformations and certain “special” transformations. For example, it would be interesting to explore weak keys, semi-weak keys, light keys (keys with a low density of ones), heavy keys (keys with a high density of ones), and pairs of related keys (*e.g.* keys that differ in one bit and keys that are complements of each other).

3.4 Extended Message Space Closure Tests

For any experiment that uses the cycling closure test, perform the cycling closure test with an extended message space that consists of the Cartesian product M^l of the original message space, for some small integer l .¹⁰

The closure test works by computing a statistical lower bound on the length of $\langle T \rangle$ -orbit(x_0), which, in turn, yields a lower bound on the order of $\langle T \rangle$. Limits on the lower bounds achievable by this test are imposed both by the number of steps the test is carried out and by the relative sizes of the message space and key space. For all $1 \leq r \leq \sqrt{M}$, if the test is run for r steps without detecting a cycle, then with high probability $\text{order}(\langle T \rangle) \geq r^2$. To use the cycling closure test to obtain statistical lower bounds on $\text{order}(\langle T \rangle)$ greater than 2^{64} , it is necessary to perform an extended message test with $l > 1$.

3.5 Reduced Message Space Tests

Perform each of the above tests on a modified version of DES in which the message space is reduced in size. Specifically, consider DES-derived functions $\phi_k : M_r \rightarrow M_r$ on the reduced message space $M_r = \{0, 1\}^r$, where r is some small integer (say, $r = 8$) and ϕ_k is defined as follows. For each key $k \in K$, define ϕ_k by $\phi_k = \pi_2 T_k \pi_1$, where $\pi_1 : M_r \rightarrow M$ is an injection and $\pi_2 : M \rightarrow M_r$ is a projection. (For example, π_1 might fix the first 56 DES input bits to 0, and π_2 might take only the last 8 DES output bits.)

¹⁰In the extended message space closure test, the pseudo-random function ρ maps M^l into K .

No.	Experiment	Leader length	Cycle length	p_G	p_R
1	Closure	$\approx 2^{25}$	$\approx 2^{33}$	$\leq 10^{-193}$	≥ 0.17
2	Closure	$\approx 2^{30}$	$\approx 2^{33}$	$\leq 10^{-264}$	≥ 0.09
3	Closure	$\approx 2^{31}$	$\approx 2^{30.5}$	$\leq 10^{-41}$	≥ 0.68
4	Extended closure	(no cycle in 2^{34} steps)		$\leq 10^{-889}$	$\geq 1 - 10^{-18}$
5	Purity	$\approx 2^{31.5}$	$\approx 2^{30}$	$\leq 10^{-81}$	≥ 0.57
6	Purity	$\approx 2^{30}$	$\approx 2^{31}$	$\leq 10^{-94}$	≥ 0.42
7	Small subgroup	0	$\approx 2^{33}$	*	$\leq 10^{-9}$
8	Orbit	(no cycle in 2^{38} steps)		*	$\geq 1 - 10^{-8}$

* Depends on hypothesized group structure.

Table 1: Summary of DES experiments, May–August, 1985. (The numbers p_G and p_R are the conditional probabilities of the experimental evidence under the hypotheses “DES is closed (pure)” and “Each DES transformation was drawn at random from the symmetric group on \mathcal{M} ” respectively.)

Studying reduced message space versions of DES is useful for two reasons. First, it is one way to look for structures that may be present on subsets of the message space. Second, by sufficiently restricting the message space, it is possible to write down a complete description of the action of particular transformations on the reduced message space.

4 Experimental Results and Conclusions

This section summarizes our experimental results and discusses two interesting structural findings.

4.1 Summary of Experimental Results

During May to August 1985, we performed eight cycling experiments covering five different algebraic tests. Specifically, we performed three closure tests, one extended message space closure test, two purity tests, one small subgroup test using two of the weak keys, and one orbit test.¹¹ These experiments gathered overwhelming statistical evidence that DES is neither pure nor closed and that the size of the group generated by DES is at least 2^{58} . Table 1 summarizes our experimental results.

As one test of correctness, we ran a software implementation of the cycling closure test for 30,000 steps. The software and hardware implementations agreed on all values. As a second test of correctness, we repeated experiments 1 and 2 and obtained identical results. We invite the interested reader to verify our results using the detailed experimental data found in appendix A.

In experiment 7, we applied the small subgroup test to the transformations represented by the two weak keys that consist respectively of all zeros and all ones. Since each of the weak transformations is self inverse, we implemented this test as an orbit test using the composition of the weak transformations. This experiment produced a short cycle of about 2^{33} steps, which would be unusual (probability less than 10^{-9}) if the tested permutation were chosen at random from $S_{\mathcal{M}}$.

¹¹We also performed one trial of a reduced message space closure test that detected no algebraic weaknesses.

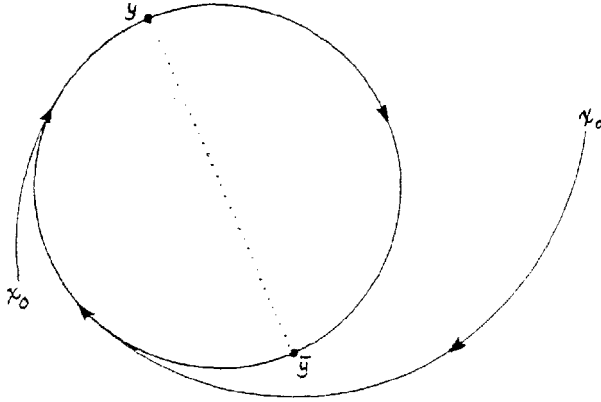


Figure 1: Results of experiments 1 and 2. Starting at different initial messages, both pseudo-random walks entered the same cycle. Every message on the cycle is the bitwise complement of the corresponding message halfway around the cycle.

4.2 Two Structural Findings

Although most of our experimental results are consistent with the hypothesis that DES acts like a set of randomly chosen permutations, three experiments did yield interesting regularities. One regularity is a result of the well-known complementation property;¹² the other involves a newly discovered property of the weak keys. We will now explain these structural findings.

4.2.1 Complementation and Drainage Properties

In the first two experiments, we performed two independent trials of the cycling closure test. Each of these experiments used the “identity” next key function—the function $\rho: \mathcal{M} \rightarrow \mathcal{K}$ that removes each of the eight parity bits. These two experiments produced two interesting findings. First, each of the pseudo-random walks drained into the same cycle. Second, each point on the cycle was the bitwise complement of the corresponding point exactly halfway around the cycle. Figure 1 illustrates these findings.

The first finding is explained by the fact that, for the graph of a randomly chosen function, most points on the graph will probably drain into the same cycle. See [HeR82] for one analysis of this phenomenon.

The second finding is a consequence of DES’s complementation property and the fact that the identity next key function also has a complementation property (for all messages x , $\rho(\bar{x}) = \overline{\rho(x)}$). The cycling closure test computes a pseudo-random walk x_0, x_1, \dots , where $x_{i+1} = T_{\rho(x_i)}(x_i)$, for $i \geq 1$. If $x_i = \bar{x}_j$ for any $i > j$, then it would follow that

$$x_{i+1} = T_{\rho(x_i)}(x_i) = T_{\rho(\bar{x}_j)}(\bar{x}_j) = \overline{T_{\rho(x_j)}(\bar{x}_j)} = \overline{T_{\rho(x_j)}(x_j)} = \bar{x}_{j+1}. \quad (2)$$

Therefore, by induction, $x_{i+h} = \bar{x}_{j+h}$ for all $h \geq 0$. This situation arises whenever some $x_i = \bar{x}_j$ before any $x_i = x_j$ with $i > j$, which will happen for about half of all initial messages.

¹²For every key k and every message x , $T_k(x) = \overline{T_k(\bar{x})}$ [DaP84].

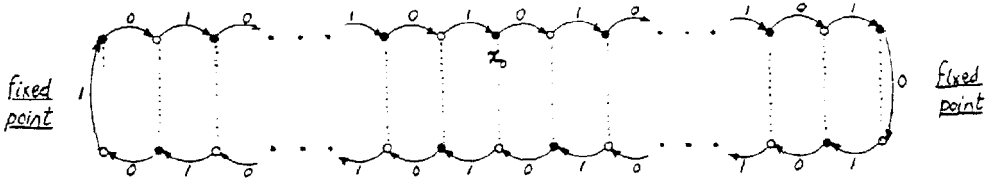


Figure 2: Results of experiment 7. (Filled circles denote the messages x_i on the $T_{1...1}T_{0...0}$ -orbit of an initial message x_0 . Unfilled circles denote intermediate values $T_{0...0}(x_i)$. Dotted lines link identical messages.)

4.2.2 Fixed Points of the Weak Keys

In experiment 7, we computed the orbit of a message under the composition of the two weak keys that consist respectively of all zeros and all ones. Although each weak key is self-inverse, we did not expect the composition to produce short orbits. Much to our surprise, we detected a cycle of length less than 2^{33} . We presented this finding at the Crypto 85 conference and sought a simple explanation.

After some thought, Don Coppersmith suggested that we had encountered fixed points of the weak keys, i.e., messages x for which $T_{1...1}(x) = x$ or $T_{0...0}(x) = x$. Since each weak key yields 16 identical round keys, for each weak key, a fixed point results whenever DES's L and R registers agree after eight rounds. Since the middle L and R registers are equal with probability about $1/2^{32}$, there should be about 2^{32} fixed points for each of the four weak keys. Hence, by 2^{33} steps, it was likely that we had encountered a fixed point. Figure 2 illustrates the effect of the fixed points on the walk in the message space and explains why a cycle resulted.

After the conference, we found the fixed points and thus confirmed Coppersmith's hypothesis (see appendix). To the best of our knowledge, these fixed points are the first published in the open literature. These fixed points further illustrate the deficiencies of the weak keys.

Coppersmith also suggested that the algebraic structure detected in experiment 7 can be used to prove strong lower bounds on the size of the group generated by DES. Experiment 7 computed the length, l , of the g -orbit of x_0 , where $g = T_{1...1}T_{0...0}$ is composition of two DES transformations and x_0 is the initial message. Since l divides the order of g , it follows that l divides the order of the group generated by DES. Therefore, if experiment 7 were repeated r times with different initial messages, and if these experiments yielded orbit lengths l_1, l_2, \dots, l_r , then $\text{lcm}(l_1, l_2, \dots, l_r)$ would be a lower bound on the order of the group generated by DES. We have not yet extended our results in this direction.

Acknowledgments

We would like to thank several people who contributed to this paper. Leon Roisenberg helped out with the design and construction of our special-purpose hardware. As part of his bachelor's thesis, John Hinsdale wrote the C software used by our host IBM personal computer to carry out the cycle-detection algorithm. We are also grateful to László Babai, Don Coppersmith, and Gary Miller for helpful comments. In addition, we would like to thank the Functional Languages and Architectures Research Group of the MIT Laboratory for Computer Science for use of their

hardware laboratory during the construction and testing of our special-purpose hardware.

References

- [Bet82] Beth, Thomas, *ed.*, *Cryptography, Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, March 29–April 2, 1982*, Springer (Berlin, 1983).
- [Bov80] Bovey, J. D., "An approximate probability distribution for the order of elements of the symmetric group," *Bull. London Math Society*, **12** (1980), 41–46.
- [BoW77] Bovey, John; and Alan Williamson, "The probability of generating the symmetric group," *Bull. London Math Society*, **10** (1978), 91–96.
- [Car56] Carmichael, Robert D., *Introduction to the Theory of Groups of Finite Order*, Dover (New York, 1956).
- [CRS82] Chaum, David; Ronald L. Rivest; and Alan T. Sherman, *eds.*, *Advances in Cryptology: Proceedings of Crypto 82*, Plenum Press (New York, 1983).
- [DaP84] Davies, Donald W.; and W. L. Price, *Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer*, John Wiley (Chichester, England, 1984).
- [Dav82] Davies, Donald W., "Some regular properties of the DES," in [CRS82], 89–96.
- [DaP82] Davies, Donald W.; and G. I. P. Parkin, "The average size of the key stream in output feedback mode," in [CRS82], 97–98.
- [DaP82a] Davies, Donald W.; and G. I. P. Parkin, "The average size of the key stream in output feedback encipherment," in [Bet82], 263–279.
- [Dix69] Dixon, John D., "The probability of generating the symmetric group," *Math Zentrum*, **110** (1969), 199–205.
- [FIPS77] "Data Encryption Standard," National Bureau of Standards, Federal Information Processing Standards Publications No. 46 (January 15, 1977).
- [FIS80] "DES modes of operations," Federal Information Standards Publication No. 81 (December 1980).
- [Gai77] Gait, Jason, "A new nonlinear pseudorandom number generator," *IEEE Transactions on Software Engineering*, **SE-3** (September 1977), 359–363.
- [Har59] Harris, Bernard, "Probability distributions related to random mappings," *Annals of Math. Statistics*, **31** (1959), 1045–1062.
- [Hel76] Hellman, Martin E., *et al.*, "Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard," technical report SEL 76-042, Information Systems Laboratory, Stanford Univ. (November 1976).
- [HeR82] Hellman, Martin E.; and Justin M. Reyneri, "Distribution of Drainage in the DES," in [CRS82] (1982), 129–131.

- [Jue82] Jueneman, Robert R., "Analysis of certain aspects of output-feedback mode," in [CRS82] (1982), 99–127.
- [KRS85] Kaliski, Burton S., Jr.; Ronald L. Rivest; and Alan T. Sherman, "Is the Data Encryption Standard a Group?" *Proceedings of Eurocrypt 85*, Springer, to appear.
- [Knu69] Knuth, Donald E., *Seminumerical Algorithms in The Art of Computer Programming*, vol. 2, Addison-Wesley (1969).
- [MeH81] Merkle, Ralph C.; and Martin E. Hellman, "On the security of multiple encryption," *CACM*, **24** (July 1981), 465–467.
- [MeM82] Meyer, Carl H.; and Stephen M. Matyas, *Cryptology: A New Dimension in Computer Data Security*, John Wiley (New York, 1982).
- [PuW68] Purdom, Paul W.; and J. H. Williams, "Cycle length in a random function," *Transactions of the American Mathematics Society*, **133** (1968), 547–551.
- [Rot78] Rotman, Joseph J., *The Theory of Groups: An Introduction*, Allyn and Bacon (Boston, 1978).
- [Sha49] Shannon, Claude E., "Communication theory of secrecy systems," *Bell System Technical Journal*, **28** (October 1949), 656–715.
- [SSY82] Sedgewick, Robert; Thomas G. Szymanski; and Andrew C. Yao, "The complexity of finding cycles in periodic functions," *Siam Journal on Computing*, **11** (1982), 376–390.
- [ShL66] Shepp, L. A.; and S. P. Lloyd, "Ordered cycle lengths in a random permutation," *Transactions of the American Mathematics Society*, (February 1966), 340–357.
- [Tuc78] Tuchman, W. L., talk presented at National Computer Conference, (June 1978).
- [Wie64] Wielandt, Helmut, *Finite Permutation Groups*, Academic Press (New York, 1964).

A Detailed Descriptions of Experiments

This appendix presents nine tables that describe in detail the cycling experiments we carried out during summer 1985. The first table defines the pseudo-random next key function used in several of the experiments. The remaining eight tables—one for each experiment—list all relevant experimental parameters together with important checkpoints encountered during the experiments.

A.1 Notation

In the body of the abstract, we defined the key space of DES to be the set $\mathcal{K} = \{0, 1\}^{56}$. Most DES implementations, however, nominally treat each key as a string of 64 bits, where every eighth key bit is a *parity bit* which is ignored. In this appendix, we too shall specify keys and messages as 64-bit strings, described in hexadecimal notation. To do this, it is convenient to introduce the DES function $\hat{T}: \hat{\mathcal{K}} \times \mathcal{M} \rightarrow \mathcal{M}$ that operates on the nominal key space $\hat{\mathcal{K}} = \{0, 1\}^{64}$.

A.2 Next Key Functions

The cycling closure test depends on a function $\rho: \mathcal{M} \rightarrow \mathcal{K}$ to compute the next key from the current message. We will now describe the two particular *next key functions* that we used during our experiments. We will define each next key function in terms of its related function $\hat{\rho}: \mathcal{M} \rightarrow \hat{\mathcal{K}}$.

Each next key function operated in a byte-by-byte fashion using a byte substitution table (1 byte = 8 bits). For any $0 \leq i \leq 7$ and any $x \in \mathcal{M}$, let $x^{(i)}$ denote the i^{th} byte of x . For each $0 \leq i \leq 7$, we computed $\hat{\rho}(x)^{(i)} = S(x^{(i)})$, for some byte substitution table $S: \{0, 1\}^8 \rightarrow \{0, 1\}^8$.

In experiments 1 and 2, we chose S to be the identity function. In the other cycling closure experiments, we used the byte substitution table given by table 2.¹³ This table was designed so that each entry has odd parity and such that each entry appears exactly twice. The table was generated using the random number generator in the C library on our IBM PC.

For the experiments that used the extended message space \mathcal{M}^2 , we computed $\hat{\rho}(x)^{(i)} = S(x^{(2i)})$ using the substitution table given in table 2.

A.3 Selection of Experimental Parameters

We chose initial messages and keys in a variety of *ad hoc* ways. Some we selected in an obviously deterministic manner (e.g., $x_0 = 0123456789ABCDEF$). Others are related to the authors' social security numbers or other personal data. The rest we generated using DES and MACSYMA.

A.4 Detailed Experimental Results

Tables 3–10 list the detailed results of our cycling experiments.

¹³The substitution table is used as follows. To substitute any byte B , consider the representation of B as two hexadecimal digits. Select the table entry whose row is given by the first digit and whose column is given by the second digit.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	3E	46	B6	26	AE	F8	2A	AE	CE	57	E6	98	07	5D	92	2C
10	FE	58	EF	CD	F7	76	2F	91	8F	0E	DO	07	B0	73	51	
20	20	6E	76	B3	86	9D	16	01	31	EF	D3	8F	D6	40	2A	FB
30	01	C7	C7	19	F7	31	A2	62	9E	B9	DA	D9	34	85	19	D9
40	61	A8	3D	BO	OE	79	C2	BC	52	04	37	FD	6E	85	FB	BA
50	DF	C8	6D	13	43	1C	0B	4A	89	83	E3	20	4F	A7	BA	3B
60	80	DO	67	EA	7F	A8	C8	43	79	6D	1A	4C	A7	CB	86	23
70	5B	02	C2	4C	58	38	FE	CE	B9	1C	15	A4	25	29	1A	15
80	C1	98	7F	4A	64	57	97	32	26	F2	E5	91	D6	E9	6B	F4
90	4F	80	67	DF	F1	BF	B3	B5	3E	E5	7A	EC	A1	B5	92	29
AO	10	DC	97	46	94	CB	49	6B	10	45	3B	F2	E6	FD	B6	BC
BO	40	OD	1F	AD	52	BF	62	23	61	49	E0	0D	08	CD	E3	C4
CO	68	1F	9E	E9	FB	7C	13	75	8A	89	04	5D	6E	DC	54	D5
DO	EA	F1	9D	F4	94	75	D3	70	8C	54	AB	2C	D5	02	98	7A
EO	3D	5B	25	8A	A1	38	8C	EC	70	9B	A4	45	64	51	AB	7C
FO	C1	AD	34	C4	EO	A2	68	83	16	08	DA	32	73	37	0B	5E

Table 2: Byte substitution table for pseudo-random next key function.

Experiment 1		$x_{i+1} = \tilde{T}_{x_i}(x_i)$
i	x_i	Note
0	0123456789ABCDEF	
34,293,588	BOFDED3BDODD918C	end of leader
34,293,589	AE5630A0E971B5E8	start of cycle
2,030,556,568	12B67D3796106D30	quarter cycle
4,026,819,547	51AACF5F168E4A17	half cycle
6,023,082,526	ED4982C869EF92CF	three-quarters cycle
8,019,345,504	A032CEOD3F438EFE	end of cycle
8,019,345,505	AE5630A0E971B5E8	restart of cycle

Table 3: Closure experiment with identity next key function. Cycle length $7,985,051,916 \approx 2^{33}$; leader length $34,293,589 \approx 2^{25}$.

Experiment 2		$x_{i+1} = \tilde{T}_{x_i}(x_i)$
i	x_i	Note
0	121502850B020664	
1,389,523,413	48BB5C9F86CD285A	end of leader
1,389,523,414	AFF50E97653421BF	start of cycle
5,152,082,299	AE5630A0E971B5E8	experiment 1 intersection
9,374,575,329	FBOA1398E92D1473	end of cycle
9,374,575,330	AFF50E97653421BF	restart of cycle

Table 4: Closure experiment with identity next key function. Cycle length $7,985,051,916 \approx 2^{33}$; leader length $1,389,523,414 \approx 2^{30}$.

Experiment 3		$x_{i+1} = \hat{T}_{\hat{p}(x_i)}(x_i)$
i	x_i	Note
0	6036222982B03104	
2,138,241,978	68955F4BF000A6E0	end of leader
2,138,241,979	C9DB8E7169CCF272	start of cycle
3,706,679,992	433B74E2CB18DDFD	end of cycle
3,706,679,993	C9DB8E7169CCF272	restart of cycle

Table 5: Closure experiment with pseudo-random next key function. Cycle length 1,568,438,014 $\approx 2^{30.5}$; leader length 2,138,241,979 $\approx 2^{31}$.

Experiment 4		$x_{i+1} = \hat{T}_{\hat{p}(x_i)}(x_i), x_i \in M^2$
i	x_i	Note
0	4C957F303AC4D08B 63E15C9C7A398042	
4,294,967,296	2C173869EAF8804B 767469BB19B26D8A	2^{32} iterations
8,589,934,592	4349368A49700D3B 55FC02F8848BC64F	2^{33} iterations
12,884,901,888	55D1292F5D99B268 C30AB80FF3B03D08	$3 \cdot 2^{32}$ iterations
17,179,869,184	4A224C65B8A48DEB 00C7DOCA64C4B240	2^{34} iterations

Table 6: Extended closure experiment with pseudo-random next key function. No cycle detected in 2^{34} steps.

Experiment 5		$x_{i+1} = \hat{T}_{\hat{k}}^{-1} \hat{T}_{\hat{p}(x_i)}(x_i)$
i	x_i	Note
0	0123456789ABCDEF	
3,233,340,362	OEC45F7157BD8749	end of leader
3,233,340,363	EFE7B7112233DD88	start of cycle
4,531,729,424	CO9DFA478C3849BE	end of cycle
4,531,729,425	EFE7B7112233DD88	restart of cycle

Table 7: Purity experiment with pseudo-random next key function. Cycle length 1,298,389,062 $\approx 2^{30}$; leader length 3,233,340,363 $\approx 2^{31.5}$. Key $\hat{k} = 97778E1BC3FD8E07$.

Experiment 6		$x_{i+1} = \hat{T}_k^{-1} \hat{T}_{\chi(x_i)}(x_i)$
i	x_i	Note
0	121502850B020664	
1,366,287,307	E43D6EF9351DDB4A	end of leader
1,366,287,308	75C8C23C21EA50DA	start of cycle
5,584,675,814	FDBE1ECDF38BF3E5	end of cycle
5,585,675,815	75C6C23C21EA50DA	restart of cycle

Table 8: Purity experiments with pseudo-random next key function. Cycle length $4,218,388,507 \approx 2^{32}$; leader length $1,366,287,308 \approx 2^{30}$. Key $\hat{k} = 4D3FD0FED9A4FA9B$.

Experiment 7		$x_{i+1} = \hat{T}_{1..1}(\hat{T}_{0..0}(x_i))$
i	x_i	Note
0	0123456789ABCDEF	start of cycle
2,227,161,945	654B672D3DBC73AB	0...0 fixed point
4,454,323,890	293FD4F2C13DD94F	"hidden crossing"
5,890,012,565	3CC5B06ADEFD3CA0	1...1 fixed point
7,325,701,239	0123456789ABCDEF	restart of cycle

Table 9: Small subgroup experiment using weak keys. Cycle length $7,325,701,239 \approx 2^{33}$; leader length 0.

i	x_i	Note
17,179,869,184	B98C3A67CD6F8267	2^{34} iterations
34,359,738,368	632509BC9F57DF8A	2^{35} iterations
51,539,607,552	ED4B06ABBF5515FB	$3 \cdot 2^{34}$ iterations
68,719,476,736	2C84263510AED34	2^{36} iterations

Table 10: Orbit experiment. No cycle detected in 2^{35} steps. Key $\hat{k} = 116E0B8278AEC431$.

A LAYERED APPROACH TO THE DESIGN OF PRIVATE KEY CRYPTOSYSTEMS

T. E. Moore and S. E. Tavares
Department of Electrical Engineering
Queen's University
Kingston, Ontario, Canada. K7L 3N6

ABSTRACT

This paper presents a layered approach to the design of private key cryptographic algorithms based on a few strategically chosen layers. Each layer is a conceptually simple invertible transformation that may be weak in isolation, but makes a necessary contribution to the security of the algorithm. This is in contrast to algorithms such as DES which utilize many layers and depend on S-boxes that have no simple mathematical interpretation. A property called transparency is introduced to deal with the interaction of layers and how they must be selected to eliminate system weaknesses.

Utilizing this layered approach, a private key cryptographic algorithm consisting of three layers is constructed to demonstrate the design criteria. The algorithm has an adequate key space and valid keys can be easily generated. The design is based on a symmetrical layered configuration, which allows encryption and decryption to be performed using the same algorithm. The algorithm is suitable for VLSI implementation. Some statistical tests are applied to the algorithm in order that its cryptographic performance can be evaluated. The test results and attempts at cryptanalysis suggest that the three-layered algorithm is secure.

1. HISTORY OF LAYERING

The concept of layering cryptographic transformations to produce stronger ones was first suggested by Shannon [14] using substitution and permutation operations as layers. This idea was introduced in 1949 as product ciphers, which made it possible to generate strong cryptosystems by concatenating weak transformations. The 'Lucifer' cipher, developed at IBM by Feistel [6] embodies this approach by alternately applying substitutions and permutations.

A well-known example of an existing private key cryptographic algorithm is the Data Encryption Standard (DES) [3]. The DES algorithm consists of many layers exemplifying the strength of a layering technique. Although DES has been adopted as an encryption standard, it

has been subjected to a great deal of criticism and suspicion [4, 7]. Some features of DES, such as the design of the S-boxes for example, are not well understood and instead of trusting in a system which is difficult to analyze, a user may choose a simpler system that can be understood.

Layered encryption has also been explored in the broadcast environment by Spencer and Tavares [15]. Only a few layers were employed in this particular application, each of an arithmetic nature. This is in contrast to layers such as those used in the DES algorithm.

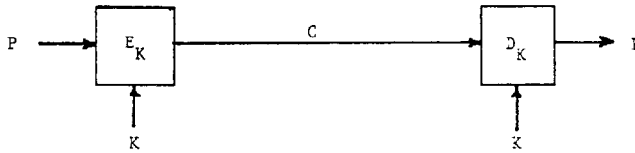
2. OVERVIEW OF LAYERING

In order that the concepts of layered encryption systems can be examined, the basic characteristics of conventional systems are stated. The components necessary in all cryptographic systems are a plaintext space P , ciphertext space C , key space K , a set of enciphering transformations E , and a corresponding set of deciphering transformations D .

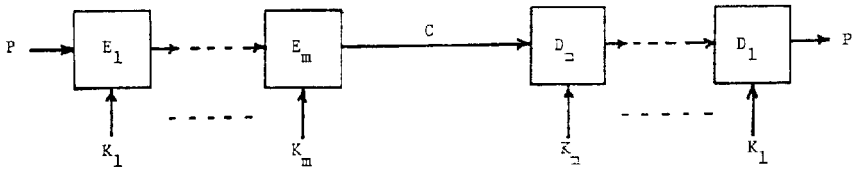
Unlike conventional systems, a layered cryptosystem has several concatenated enciphering transformations for encryption and the same number of deciphering transformations concatenated together for decryption. An m -layered cryptosystem is composed of a plaintext space P , ciphertext space C , a set of m key spaces K_1, \dots, K_m , m sets of enciphering transformations E_1, \dots, E_m , and m corresponding sets of deciphering transformations D_1, \dots, D_m . Schematic diagrams of these two types of cryptosystems are given for comparison in Figure 1.

There are three basic assumptions important to the functionality of layered cryptosystems. The first is that the set of individual layer keys k_1, \dots, k_m used for encryption are kept secret from unauthorized users. Secondly, each layer is a simple invertible transformation which may be weak cryptographically in isolation, but makes a necessary contribution to the security of the entire system. Lastly, the interlayer results of the enciphering and deciphering transformations are not accessible to unauthorized users. All discussions dealing with layered encryption in this paper apply only to private key cryptosystems.

It is important here to clarify that any layer by itself is not secure, given access to its input and output values. Nothing is gained by layering if interlayer results are an allowable resource to a cryptanalyst. It is a reasonable assumption to consider the inter-



(a) Cryptographic System



(b) M-Layered Cryptographic System

FIGURE 1: Comparison of Conventional and Layered Cryptosystems.

layer results as unattainable resources. Unlike plaintext and ciphertext, interlayer values are only transient results which are never stored or accessed at any time by legitimate users of the system. The only manner in which they may be obtained is if an intruder can tap and monitor the hardware between the layers. Physical security can always be employed if monitoring is a possible threat. The remainder of this discussion assumes that interlayer results are never accessible and are adequately protected.

3. THE LAYERED APPROACH

3.1 Layer Selection Criteria

By adopting a few strategically chosen layers, a layered approach can be utilized to design private key cryptosystems. Before a mathematical transformation may be classified as a layer, it must conform to the following specifications:

- a) a layer must be well defined in a mathematical sense while remaining simple in concept

- b) it must have an adequate key space with easily generated keys and inverse keys
- c) be efficient in terms of time and space
- d) be easy to program for software simulation and implementation and
- e) be suitable for VLSI design and implementation.

It is plausible that if each individual layer in a layered cryptosystem meets these requirements, then the synthesized layered algorithm will conform to them as well.

3.2 Layer Interaction and Transparency

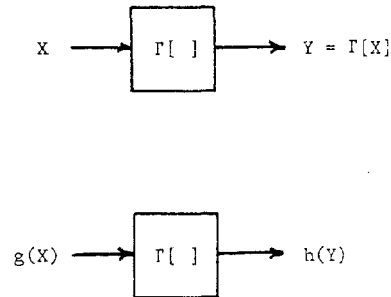
With the layer selection criteria established, it becomes necessary to develop additional guidelines for concatenation of layers. An important consideration in concatenating layers to synthesize a complete algorithm is the problem of layer interaction. There is an obvious disadvantage to concatenate two layers which can each be compromised on an individual basis by the same attack.

A concept is introduced here which helps to deal with layer interaction and is defined as layer transparency. To define transparency, consider the transformation $r []$ of Figure 2 which maps X into Y , where X and Y are n -bit vectors. Let $g(X)$ be the result of a simple operation $g(\cdot)$ on the input X . If $g(X)$ is mapped to $h(Y)$ by $r []$, where $h(\cdot)$ is also a simple operation, then it is said that $r []$ is transparent to $g(\cdot)$, and that $g(\cdot)$ is a transparency of $r []$. In this discussion, it should be noted that $g(\cdot)$ and $h(\cdot)$ may be the same operation. As an example, $g(X)$ could be a cyclic shift of X by one bit and $h(Y)$ a cyclic shift of Y by t bits, where $1 \leq t \leq n-1$. If $t = 1$, then the two operations $g(\cdot)$ and $h(\cdot)$ would be identical.

As a general rule, two adjacent layers in a layered cryptographic algorithm should not have common transparencies. In addition, it is desirable that all layers in a cryptosystem do not share many of the same transparencies.

3.3 Buffers

The problem of selecting various useful transformations that strictly follow the two transparency rules may not be simple. What is required are simple operations to isolate the main layer transformations. As an example, two nearly compatible transformations may be suitable as adjacent layers except for a single common transparency. If a simple operation can be found that does not preserve this common transparency, then it can be inserted between the two layers. The resultant



$\Gamma[]$ - invertible transformation

X - n -bit input vector

Y - n -bit output vector

$g(\cdot)$, $h(\cdot)$ - simple operations

FIGURE 2: Illustration of Transparency.

new transformation of a simple layer sandwiched between two main layers is no longer hampered by the transparency. The simple operations in question are defined as 'buffers', and for simplicity they can be considered as another layer in the layered cryptosystem. However, buffers differ from the main layers in that they do not possess a key space.

There are two types of buffers defined by their position relative to the main transformations. The first type are positioned before the first and after the last layers. This buffer type is defined as an 'outer buffer'. In a cryptosystem of only a few layers, it is critical that a cryptanalyst not be allowed to probe the outer layers using strategically selected inputs. Knowledge of the transparencies of the first layer for example, can be utilized in such a manner as to derive the result of this transformation without actual knowledge of its key. Hence, for the given strategic input, the first layer is effectively by-passed leaving a weakened algorithm to compromise.

It is realized that a constant operation is not suitable for an outer buffer. Since we assume that every feature of the cryptographic algorithm will be public knowledge, except for the key of course, a cryptanalyst can derive the result of any constant operation and have direct access to the outer layers as before. It is thus necessary that outer buffers be computed from key-dependent operations so that

the result of a given buffer operation cannot be determined without knowledge of the keys. For a given key set, this may be accomplished by computing the buffers from a single one-way function of the layer keys. Hence, actual inputs to the first main layer cannot be derived, preventing effective chosen-plaintext attacks.

The second type of buffers are positioned between two main layers. These buffers are defined as 'interlayer buffers' and their purpose is to prevent the preservation of transparencies that exist in common with two adjacent main transformations.

In contrast to an outer buffer, the input to any interlayer buffer is never directly accessible, making it unnecessary for interlayer buffers to be key-dependent operations. Further, it is preferable if the interlayer buffers are key-independent operations as they would not require any pre-computation for a given key set.

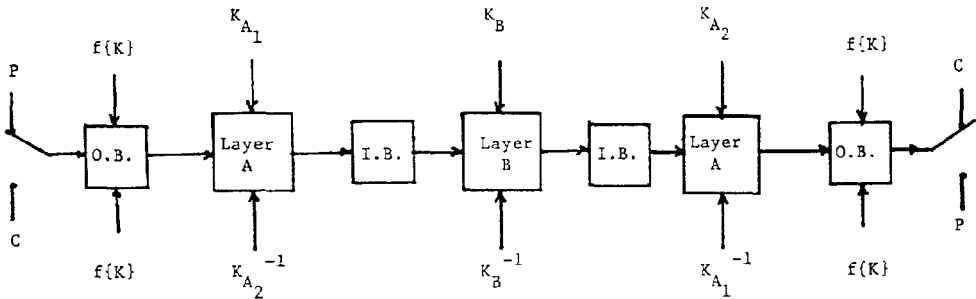
3.4 Additional Considerations

In a system where all main layers and buffers are linear, the system transformation may be represented equivalently by a simplified linear operation. An attack based on the principle of superposition can be utilized to compromise a linear cryptosystem. It is thus necessary to ensure that the overall system transformation for the layered algorithm is nonlinear. This can be accomplished by selecting one of the main layers as a nonlinear transformation.

A second consideration when dealing with layer concatenation is symmetry. Carefully selecting the layers in a symmetrical configuration will allow the encryption and decryption functions to be performed using the same algorithm. A schematic diagram of a symmetrical layered configuration is given in Figure 3. For this 3-layered example illustrated in the figure, the essential nonlinear transformation can be either Layer A or Layer B.

In order to facilitate the symmetry in Figure 3, several conditions must be satisfied. First the two outside layers must be selected as identical transformations. In practice, different keys would be used for these two layers to keep the system key space as large as possible. The next requirement for total symmetry is that the two outer buffers must be identical operations. The interlayer buffers must also meet this requirement. The relative positions of these buffers are clearly illustrated in Figure 3. The last requirement is that the outer buffers must be their own inverse operations. The interlayer buffers must also fulfill this requirement.

ENCRYPTION



DECRYPTION

P - plaintext
 C - ciphertext
 $K = \{K_{A1}, K_B, K_{A2}\}$
 = encryption key set
 $K^{-1} = \{K_{A2}^{-1}, K_B^{-1}, K_{A1}^{-1}\}$
 = decryption key set

$O.B.$ - Outer Buffer
 $I.B.$ - Interlayer Buffer
 $f(K)$ - one-way function

FIGURE 3: Symmetrical Layered Configuration.

With these conditions included in a symmetrical layered configuration, total algorithm symmetry is obtained. As shown in Figure 3, if the encryption key set is $\{K_{A1}, K_B, K_{A2}\}$, then the corresponding decryption key set is $\{K_{A2}^{-1}, K_B^{-1}, K_{A1}^{-1}\}$. In this notation, K_{A1}^{-1} represents the mathematical inverse (decryption key) of K_{A1} for the transformation of Layer A. A benefit resulting from designing a symmetrical algorithm is the reduction in the amount of chip area needed to incorporate both encryption and decryption in a single chip VLSI implementation.

3.5 Summary of Approach

The important concepts pertinent to the layered design approach of cryptographic algorithms were presented. Selection criteria for transformations were established and the concept of transparency was introduced to resolve the problem of layered interaction. System

transparencies can be eliminated by carefully selecting transformations with specified properties, and by utilizing specially designed buffers. The presence of at least one nonlinear operation is essential to the security of the algorithm. The essential nonlinearity can be accommodated by selecting a nonlinear transformation as one of the layers. A symmetrical layered configuration has several advantages, but is not necessary for constructing a secure system. By selecting certain transformations and concatenating them using the established criteria in this section, it may be possible to synthesize a secure cryptosystem.

4. DESIGN OF A LAYERED CRYPTOGRAPHIC ALGORITHM

Before presenting the following discussion, it is important to clarify that the algorithm given here is not intended to represent an unbreakable cryptosystem. It is simply given here in order to illustrate the structured approach to designing cryptographic algorithms given in the previous section.

Utilizing the layered approach given in Section 3, a private key cryptographic algorithm has been designed using the exact configuration given in Figure 3. The Layer A transformations have been selected as linear transformations and Layer F as the essential nonlinear transformations.

4.1 Nonlinear Layer

There are a number of nonlinear transformations that have been used in cryptographic applications. For reasons of dependability and reputation as a strong algorithm, the RSA algorithm [13] was examined for possible foundations for a nonlinear transformation. On that basis, modular exponentiation was chosen to represent the nonlinear layer. A modulus of $2^n - 1$, for n -bit block encryption, was chosen as an appropriate modulus for this transformation. In this discussion, n is a power of 2, for reasons which will become evident. There are two reasons for choosing this particular modulus. First, the integer $2^n - 1$ is a product of r distinct prime numbers (at least as far as $n = 64$). This is an extension of the two prime case used with the RSA modulus. The second reason is an implementation feature of $2^n - 1$ in that actual division is not required to perform modulo reduction by $2^n - 1$. This will become clear in Section 5 where implementation considerations are discussed.

To summarize, the following nonlinear transformation is used as Layer

B in the symmetrical layered configuration of Figure 3.

$$Y = 2^{n-1}, \quad \text{if } X = 2^{n-1} \\ = X^{K_B} \bmod 2^{n-1}, \quad \text{otherwise.}$$

where

X is the n-bit input

Y is the n-bit output

K_B is the key for Layer B

and

$$n = 2^m$$

Ordinarily C and 2^{n-1} are equivalent modulo 2^{n-1} , and hence both of these inputs would produce an all zero binary output. The conditional equality in the above definition is necessary to resolve this situation.

In order that we may decrypt correctly, an inverse K_B^{-1} must exist such that

$$X^{K_B} * K_B^{-1} \bmod 2^{n-1} = X.$$

This relation can be satisfied for any modulus that is a product of r distinct primes, if the following relationship is true [2]

$$K_B * K_B^{-1} = 1 \bmod \phi(2^{n-1})$$

where $\phi(\cdot)$ is the Euler totient function. The above relation reduces to the property that K_B must be chosen relatively prime to $\phi(2^{n-1})$.

Blakley and Borosh [2] recognized that transformations of this type always have a certain number of inputs that are mapped to themselves, defined as unconcealed inputs. For this particular transformation, this phenomenon may be represented mathematically as

$$X^{K_B} \bmod 2^{n-1} = X.$$

To minimize the number of unconcealed inputs, it is required that K_B be chosen under the following additional constraint:

$$\text{GCD}[K_B - 1, \text{LCM}(p_1 - 1, \dots, p_r - 1)] = 2$$

where GCD is the Greatest Common Divisor

LCM is the Least Common Multiple

and (p_1, \dots, p_r) are the r unique prime factors of 2^{n-1} .

For exponentiation $\text{mod } 2^n - 1$, the actual minimum number of unconcealed inputs is $3^n + 1$. This minimum number can only be achieved if K_B satisfies the above relation. For a block length of $n = 64$, there are a minimum of 2188 unconcealed inputs since $2^{64} - 1$ is a product of 7 distinct prime numbers. The number of key bits generated by exponentiation modulo $2^n - 1$, with $n = 64$, is estimated to be 29.

4.2 Linear Layers

By concatenating a simple linear transformation processing specified properties with a nonlinear layer, it is plausible that a stronger transformation will result from the concatenation. A particular family of linear transformations used in cryptographic applications is modular multiplication. These transformations have been studied previously by Leung and Tavares [12] for modulus values of 2^n and $2^n - 1$. Multiplication modulo $2^n - 1$ is a fundamental transformation in a cryptographic algorithm proposed by Akl and Meijer [1].

Multiplication modulo 2^n was chosen over $2^n - 1$ for two reasons. First, this modulus is different from the modulus of the nonlinear exponentiation transformation. If each layer was some operation modulo $2^n - 1$, and ignoring the effect of any interlayer buffers, then it is possible to simplify the overall mathematical representation of the 3-layered concatenation by applying the principles of modular arithmetic [5].

The second reason is that multiplication modulo 2^n is not transparent to complements, whereas multiplication $\text{mod } 2^n - 1$ is transparent to this complement operation. To further clarify this operation, a bit-wise complement of any input produces a bit-wise complement of its corresponding output. It can be shown that both multiplication and exponentiation $\text{mod } 2^n - 1$ are transparent to complements. Thus selecting multiplication $\text{mod } 2^n - 1$ would tend to violate the layer interaction criteria established in Section 3.2. Proof of the complement transparency for exponentiation $\text{mod } 2^n - 1$ is given in Appendix A.

In summary, the linear transformations indicated by Layer A in Figure 3 may be represented analytically as:

$$Y = X * K_A \text{ mod } 2^n$$

where X is the n -bit input

Y is the n -bit output

and K_A is the key for Layer A.

In order to estimate the size of the key space for this transformation, the number of keys K_A that allow X to be recovered from Y must

be known. From elementary number theory, X has a unique inverse mod 2^n if the integers K_A and 2^n are relatively prime. The number of integers relatively prime to 2^n is $\phi(2^n) = 2^{n-1}$. Hence for $n = 64$, there are 2^{63} keys K_A that will allow successful decryption. Since the GCD ($K_A, 2^n$) must equal 1, the K_A must be an odd integer and thus valid 64-bit keys may be selected by simply setting the least significant bit of K_A to binary one.

4.3 Common Transparencies and Buffer Selection

The design of the buffers depends on the common transparencies that exist between the main transformations of the algorithm. The following is a summary of the known transparencies and weaknesses that are common to exponentiation mod 2^n-1 and multiplication mod 2^n .

- i) The all-binary zero input maps to itself in both transformations.
- ii) Both transformations are transparent to shifting; although multiplication is transparent to logical shifts, and exponentiation is transparent to a variation of cyclic shifts.
- iii) Multiplication is preserved under modular exponentiation and modular multiplication.

The transparencies and weaknesses above can be easily verified for each transformation.

Recall that the purpose of outer buffers is to inhibit an intruder from launching chosen-plaintext attacks. Outer buffers must also be key-dependent operation. For simplicity and ease of implementation, the exclusive - OR addition of a key-dependent n -bit sequence V is suitable for the outer buffers. To determine a particular value of V for a given key set, it is necessary that V be derived from a one-way function of the three keys. Therefore, the sequence V cannot be computed unless the three layer keys are known. Exclusive - OR addition is also its own inverse operation and thus satisfies the conditions needed to maintain the symmetrical layered configuration depicted in Figure 3. By coincidence, this buffer operation also eliminates the zero input mapping to itself.

The second and third common transparencies listed at the beginning of this section are left to be resolved by the interlayer buffers. It should be pointed out that the third transparency is true only when the product of the inputs in question is less than 2^n . If the product is greater than this value, then multiplication is not preserved through the concatenation of the three layers. This result stems from the fact that two different modulus values are used in the transforma-

tions.

An n -bit permutation ρ is a suitable interlayer buffer under the following constraints:

- i) ρ does not preserve shifts
- ii) ρ does not preserve multiplication
- and iii) ρ is its own inverse operation

The first two constraints rectify the second and third common transparencies and the last constraint is necessary to satisfy the symmetrical layered configuration conditions.

4.4 Summary of 3-Layered Algorithm

A block diagram summary of the 3-layered cryptographic algorithm is shown in Figure 4. The layers indicated by Layer A in Figure 3 are multiplication modulo 2^n transformations, and Layer B is the nonlinear exponentiation modulo 2^n-1 transformation. To easily distinguish between the two multiplication layers, a notation change from letters to numbers is done. The layers are labelled as 1, 2 and 3 going from left to right in Figure 4. The outer and interlayer buffers shown in the figure are as defined in Section 4.3.

If we let $T(\cdot)$ represent the overall transformation depicted in Figure 4, then the encryption operation may be represented as

$$C = T_K(P)$$

where P is the plaintext

C is the ciphertext

and $K = \{K_1, K_2, K_3\}$ is the encryption key set. Since the algorithm is symmetrical, the decryption operation may also be represented in terms of the same transformation as

$$P = T_{K^{-1}}(C)$$

where $K^{-1} = \{K_3^{-1}, K_2^{-1}, K_1^{-1}\}$ is the decryption key set. Thus, the distinguishing feature between encryption and decryption with this algorithm is the key set used in each case. The decryption keys are related to their encryption key counterparts by the following equations:

- i) $K_1 * K_1^{-1} \bmod 2^n = 1$
- ii) $K_2 * K_2^{-1} \bmod (2^n-1) = 1$
- iii) $K_3 * K_3^{-1} \bmod 2^n = 1$

Calculating the integer values of each decryption key can thus be accomplished by using Euclid's algorithm [9].

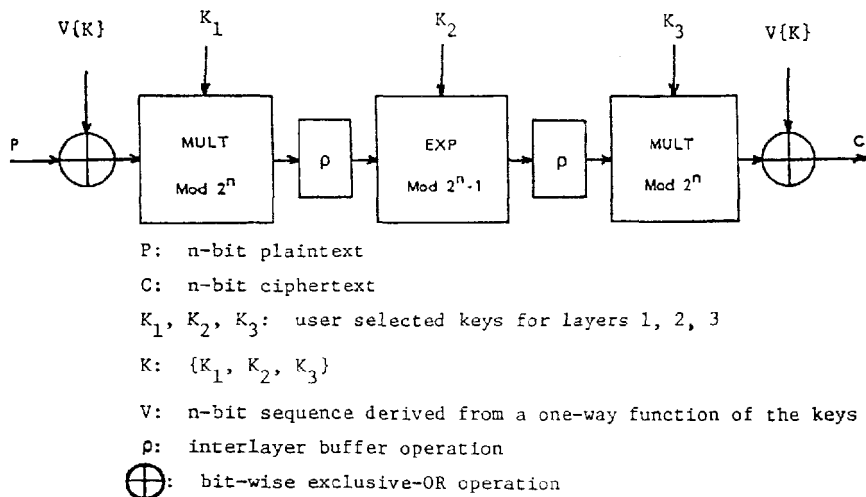


FIGURE 4: Block Diagram of the 3-Layered Cryptographic Algorithm.

5. IMPLEMENTATION CONSIDERATIONS

The discussion included in this section is intended to illustrate the relatively simple algorithms that are needed to implement the main transformations in the algorithm of Figure 4. The primary consideration of the design criteria was to facilitate a VLSI (very large scale integration) application. Pseudo-code algorithms suitable for VLSI implementation of the main transformations are contained in Appendix E. Simple shift-registers and adders are the primary components necessary to implement these algorithms.

The first pseudo-code algorithm given in Appendix B is for modular exponentiation. It uses repeated squaring and multiplication to implement exponentiation. The algorithm scans the bits of the binary representation of the exponent, starting with the least significant bit. For each bit of the exponent, a squaring operation is performed if the bit is a binary zero, squaring followed by multiplication if the current exponent bit is a binary one. All squaring and multipli-

cation operations are reduced modulo 2^n-1 , and can be implemented using the second algorithm given in Appendix P.

The second and third algorithms of Appendix P are for multiplication modulo 2^n-1 and 2^n respectively. Both utilize "shifting and adding" techniques to implement multiplication. These algorithms are efficient since actual division is not performed when modulo reducing by either 2^n or 2^n-1 .

For a modulus of 2^n , all overflow bits resulting from the repeated addition operations are simply truncated in order to modulo reduce. The overflow bits represent integer multiples of 2^n , and hence truncating these bits is equivalent to dividing by 2^n . For 2^n-1 , the overflow bits are not truncated, but are cyclicly shifted and added to the least significant bit of the result. This is equivalent to subtracting a value of 2^n-1 .

To implement the outer buffer operations in VLSI, n two-input exclusive - OR gates in parallel can be used for each buffer. Since interlayer buffers can be selected as constant permutations, they can be hard-wired in a VLSI implementation.

6. PERFORMANCE

Since the 3-layered algorithm cannot be proven secure, we must rely on certain tests and analyses to provide confidence in the algorithm. A few statistical tests have been applied to the algorithm in order to evaluate its cryptographic performance. The tests listed below were used in the evaluation:

- i) Plaintext/ciphertext Complexity Test
- ii) Avalanche Complexity Test
- iii) Bit Distribution Test
- iv) Cycle Test

The above tests were performed on a 32-bit software implementation of the algorithm. Using a VAX 11/750 computing facility, assembly language routines were written to simulate each layer.

The first two tests listed above depend on the concept of complexity. The complexity criterion [12] was used extensively for performing these statistical tests, and a measure of complexity developed by Lempel and Ziv [11] was used to evaluate the randomness properties of the algorithm. In general, the difference between any plaintext and its corresponding ciphertext should have high complexity with a high probability [12]. This complexity is referred to as plaintext/

ciphertext complexity and is measured using the first test. In addition, the difference between two ciphertexts whose corresponding plaintexts differ by a predetermined bit must also have this high complexity. Horst Feistel [6] termed this property the Avalanche Effect, and it is measured using the avalanche complexity test.

An additional test is a bit distribution test which simply counts the number of binary ones (or zeros) in the two variations of difference sequences mentioned above. Over a large sample of randomly selected plaintext, the resulting bit distribution should resemble the binomial distribution if the differences are indeed random.

The last test that was applied to the algorithm is a cycle test [10]. The purpose of this particular test is to determine if the set of permutations for the overall algorithm transformation is closed under functional composition. If the transformation is closed, then the set of transformations may generate a small group and hence contain a weakness that is vulnerable to a known-plaintext attack [8]. The cycle test that was implemented examines the orbits of plaintext messages under fixed keys which are produced by the algorithm in output-feedback mode. Although this is not the most efficient closure test [8], it was felt that this particular version of the test was the simplest and best suited for the available resources.

The results of the first three statistical tests listed at the beginning of this section indicate that the 3-layered algorithm performs well cryptographically. It appears that the algorithm in fact possesses good randomness properties. The cycle test results are inconclusive as only a few tests have been completed. Results thus far indicate that the overall transformation of the 3-layered algorithm is not closed under functional composition.

7. CLOSING REMARKS

We have presented a layered approach to designing strong cryptographic algorithms using conceptually simple mathematical transformations. Although the layers themselves are weak in isolation, they make a necessary contribution to the overall strength of the algorithm. This is a simplified approach which can reduce the complexity of designing a cryptographic algorithm.

In addition, a three-layered cryptographic algorithm has been designed using the layering technique. Although the algorithm was presented to illustrate the design criteria, it in fact appears strong and possesses several attractive features. Naturally, it is possible that

cryptanalysis could show that the algorithm is weak, or under certain conditions, may be compromised completely. In either case, the analysis would be interesting due to the simple concepts and mathematical properties inherent in the design. The layered approach would still be considered useful as it reduces the complexity of algorithm design. It also allows a designer to develop layered algorithms reasonably fast, since previously studied transformations can be chosen as layers.

ACKNOWLEDGEMENT

The authors would like to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada under Strategic Grant #G1364.

8. REFERENCES

- [1] Akl, S.G. and Meijer, H., "Two New Secret Key Encryption Algorithms", presented at Eurocrypt '85, Linz, Austria, Apr. 1985.
- [2] Blakley, G.R. and Borosh, I., "Rivest-Shamir-Adleman Public Key Cryptosystems Do Not Always Conceal Messages", *Comp. & Maths with Appls.*, Vol. 5, pp. 168-178, Pergamon Press Ltd., 1979.
- [3] "Data Encryption Standard", FIPS PUB 46, National Bureau of Standards, Washington, D.C., Jan. 1977.
- [4] Davies, D.W., "Some Regular Properties of the DES", *Advances in Cryptology: Proceedings of Crypto '82*, pp. 89-96, Plenum Press, 1983.
- [5] Denning, D.E., Cryptography and Data Security, Addison-Wesley, Reading, Mass., 1982.
- [6] Feistel, H., "Cryptography and Computer Privacy", *Sci. Am.*, Vol. 228, pp. 15-23, May 1973.
- [7] Hellman, M.E., et al., "Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard", Information Systems Lab., Dept. of Electrical Eng., Stanford Univ., 1976.
- [8] Kabiski, B.S., Rivest, R.L. and Sherman, A.T., "Is the Data Encryption Standard a Group?", presented at Eurocrypt '85, Linz, Austria, Apr. 1985.
- [9] Knuth, D., The Art of Computer Programming; Vol. 2, Semi-numerical Algorithms, Addison-Wesley, Reading, Mass., 1969.
- [10] Konheim, A.G., Cryptography: A Primer, John Wiley and Sons, New York, 1981.
- [11] Lempel, A. and Ziv, J., "On the Complexity of Finite Sequences", *IEEE Trans. on Info. Theory*, Vol. 17-22, pp. 75-81, Jan. 1976.
- [12] Leung, A.K. and Tavares, S.E., "Sequence Complexity as a Test for Cryptographic Systems", *Proceedings of Crypto '84*, pp. 468-474, Springer-Verlag, 1985.
- [13] Rivest, R.L., Shamir, A. and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Comm. ACM*, Vol. 21, pp. 120-126, Feb. 1978.
- [14] Shannon, C.E., "Communication Theory of Secrecy Systems", *Bell Syst. Tech. J.*, Vol. 28, pp. 656-715, Oct. 1949.
- [15] Spencer, M.E. and Tavares, S.E., "Layered Broadcast Cryptographic Systems", *Advances in Cryptology: Proceedings of Crypto '83*, pp. 157-170, Plenum Press, 1984.

APPENDIX A

Proof of Complement Transparency for Exponentiation Mod 2^n-1

It is required to show that a bit-wise complement of any input produces a bit-wise complement of its corresponding output for exponentiation mod 2^n-1 .

Proof: More formally, it is required to show that if

$$X^K \equiv Y \pmod{2^n-1}$$

then

$$\bar{X}^K \equiv \bar{Y} \pmod{2^n-1}.$$

where \bar{X} and \bar{Y} are the bit-wise complements of X and Y respectively and K is odd.

We can write

$$X + \bar{X} = 2^n - 1$$

$$\text{or} \quad X + \bar{X} \equiv 0 \pmod{2^n-1}$$

$$\text{or} \quad X \equiv -\bar{X} \pmod{2^n-1}$$

Squaring,

$$X^2 \equiv \bar{X}^2 \pmod{2^n-1}$$

and thus

$$X^U \equiv \bar{X}^U \pmod{2^n-1}, \quad U \text{ even}$$

and

$$\bar{X}^V \equiv -X^V \pmod{2^n-1}, \quad V \text{ odd}$$

$$\text{Let} \quad X^K \equiv Y \pmod{2^n-1}, \quad K \text{ odd}$$

$$\text{then} \quad \bar{X}^K \equiv -X^K \pmod{2^n-1}$$

$$\equiv -Y \pmod{2^n-1}$$

$$\equiv \bar{Y} \pmod{2^n-1}.$$

APPENDIX B

Pseudo-code Algorithms for Layer Transformations

In this appendix are pseudo-code algorithms for the two transformations of the 3-layered algorithm. A total of three algorithms are included as follows:

- i) Algorithm #1: Exponentiation mod 2^n-1
- ii) Algorithm #2: Multiplication mod 2^n-1
- iii) Algorithm #3: Multiplication mod 2^n

Algorithm #1 requires the use of Algorithm #2 in the form of a subroutine in order to perform the full modular exponentiation transformation. Algorithms #2 and #3 are shifting and adding based routines which include the appropriate variations necessary to perform modulo reduction.

The three algorithms are presented here purposely for a VLSI application. Each algorithm can be implemented almost entirely with shift-registers, adders, and a carry-bit function. For these algorithms, the value of 'n' is not considered variable, but is in fact a constant equal to the specified block length of the cryptosystem. Thus n will govern certain design parameters such as the size of internal registers. For notation, all input and output variables (denoted as capital letters) are regarded as n-bit integers, and the i^{th} bit position of X is expressed as X(i) in these algorithms.

ALGORITHM #1

Exponentiation mod 2^n-1

Returns: $Y = X^K \text{ mod } 2^n-1$

Input: X, K

 i = 0

 if (K(i) = 1) then

 Y = X

 else

 Y = 1

 end if

 i = 1

 do while (i < n)

 X = X * X mod 2^n-1

 if (K(i) = 1) then

 Y = X x Y mod 2^n-1

 end if

 i = i + 1

 end do

Output: Y

ALGORITHM #2

Multiplication mod $2^n - 1$ Returns: $P = A * B \bmod 2^n - 1$

Input: A, B

P = 0

i = 0

do while (i < n)

if (B(i) = 1) then

P = P + csl(i, A)

if (carry = 1) then

P = P + 1

end if

end if

i = i + 1

end do

Result: P

csl(i, A) = cyclic shift left of A by i bits

carry = carry bit function

ALGORITHM #3

Multiplication mod 2^n Returns: $Y = X * K \bmod 2^n$

Input: X, K

Y = 0

i = 0

do while (i < n)

if (K(i) = 1) then

Y = Y + lsl(i, X)

end if

i = i + 1

end do

Output: Y

lsl(i, X) = logical shift left of X by i bits

LIFETIMES OF KEYS
IN CRYPTOGRAPHIC KEY MANAGEMENT SYSTEMS

E. Okamoto and K. Nakamura
C&C Systems Research Laboratories
NEC Corporation
Miyamae-ku, Kawasaki 213 Japan

1. INTRODUCTION

Network architectures, such as System Network Architecture (SNA)^[1], have an encryption function including key management at a functional layer. SNA uses Data Encryption Standard (DES)^[2] to encrypt data and keys. A data encrypting key is encrypted with a master key and transmitted before every session. However, "the lifetime of the master key", namely, the time when the master key should be changed, is not prescribed. If the same key is used for a long time, it is probable that this secret key will be exposed.

This paper describes the lifetimes of keys. The lifetimes are the optimal key change periods, because they represent the optimal time intervals between key changes. We investigate the lifetimes of keys in two types of key distribution schemes. One scheme is the usual scheme where the data encrypting key to be used in the next session is encrypted with an upper-level key encrypting key and transmitted to the receiving side. In the other scheme this encrypted data encrypting key is encrypted again with the data encrypting key being used at the present session and transmitted to the receiving side. In both schemes, the key encrypting key may be encrypted with more upper-level key encrypting keys. In this paper, the former scheme and the latter scheme are called SCHEME 1 and SCHEME 2, respectively. The keys lifetime in SCHEME 2 is shown to be much longer than that in SCHEME 1.

In the discussion, we assume that the cryptattack is based on the simplest method, namely the exhaustive key search. It may be possible to cryptanalyze in a shorter time, using statistical characteristics of encrypted data sequences, though there has

been no such DES cryptanalysis reported so far. Hence, the lifetimes of keys described in this paper show one of the upper bounds.

Moreover, this paper deals with DES as an example. However, it utilizes only the fact that the effective key length is 56 bits, hence the discussion can be applied to other encryption algorithms.

2. KEY DISTRIBUTION SCHEMES AND THEIR CRYPTANALYSIS METHODS

Figure 1 shows two types of key distribution schemes, SCHEME 1 and SCHEME 2. In Fig. 1, E and D show the encryption transformation and the decryption transformation, respectively. M shows a message and R a register. The lowest level key, K_1 , is a data encrypting key which is called a work key. The second level key, K_2 , is used to encrypt K_1 for distribution, and so on. The highest level key, K_L , is not encrypted. It is sent via a secure channel or by a courier. Key K_L is called a master key. Every key is generated randomly at the sending side.

In SCHEME 2, when K_i is changed to K_i' , selectors $SEL_{i-1}, SEL_{i-2}, \dots, SEL_0$ select upper lines and switches $SW_{i-1}, SW_{i-2}, \dots, SW_0$ connect each decryptor output to upper lines in the figure. Hence $E_{K_i}(K_i')$ is multi-encrypted with $K_{i-1}, K_{i-2}, \dots, K_1$.

We assume that the cryptanalysis method is based on an exhaustive key search described below.

Cryptanalysis method for SCHEME 1

- 1) Obtain $C_1 = E_{K_1}(M)$, $C_2 = E_{K_2}(K_1)$, \dots , $C_L = E_{K_L}(K_{L-1})$, where $E_{K_i}(M)$ shows an encrypted message M with key K_i and C_i a cipher text.
- 2) Select a master key candidate KC_L .
- 3) Calculate lower level key encrypting key candidates $KC_{L-1} = D_{KC_L}(C_L)$, $KC_{L-2} = D_{KC_{L-1}}(C_{L-1})$, \dots , $KC_1 = D_{KC_2}(C_2)$ and message candidate $MC = D_{KC_1}(C_1)$.
- 4) If MC is the right message M, then let $K_L = KC_L$ and decode cipher texts, otherwise select another master key candidate

and go to 3)

Cryptanalysis method for SCHEME 2

- 1) Search for key K_1 from $E_{K_1}(M)$ and (partial) M . If K_1 is found, decode cipher texts until K_1 is changed.
- 2) Search for key K_2 from $E_{K_2}(K_1)$ and K_1 . If K_2 is found, decode cipher texts until K_2 is changed.

.
.
.

- L) Search for key K_L from $E_{K_L}(K_{L-1})$ and K_{L-1} . If K_L is found, decode cipher texts until K_L is changed.

There may be many K s satisfying $E_K(M)=C$ when message text M and cipher C are given. Hence, it is necessary to check $E_K(M')=C'$ with other M' and C' . If $E_K(M') \neq C'$, continue to search for K . In this paper, searching for K from $E_K(M)$ and M means to find the real K .

If a cryptanalyst stores all cipher texts into a memory, they can all be decoded after finding the keys. However, the texts are usually quite old when the keys are cryptanalyzed, because cryptanalysis requires much time. Hence, we assume that the cryptanalyst tries to obtain online real-time messages.

Disclosure rate ξ_L is defined as

$$\xi_L = \frac{\text{mean interval in which messages are disclosed}}{\text{interval in which } K_L \text{ is used}} \quad (1)$$

In general, rate ξ_L increases according to increase in the K_L length.

3. LIFETIMES OF KEYS

This section represents the lifetimes of keys as function of the number of key levels and the disclosure rate. First, the disclosure rates are derived.

1 Disclosure rate

(1) Disclosure rate for SCHEME 1

Let t_L be the time when K_L is disclosed, and T_L be the time when K_L is changed. Time t_L is a random variable. Probability density function $p(t)$ of disclosure of K_L can be expressed as,

$$p(t) = \begin{cases} \frac{1}{LA} & ; (0 \leq t \leq LA) \\ 0 & ; \text{otherwise} \end{cases} \quad (2)$$

here A is the total time in which all keys need to be investigated. In DES case, for example,

$$A = 2^{56} \cdot \tau, \quad (3)$$

where τ is the time for encrypting one block.

Messages are exposed in the period from time t_L to T_L , hence disclosure rate ξ_L is given by

$$\begin{aligned} \xi_L &= \frac{1}{T_L} \int_0^{T_L} p(t) (T_L - t_L) dt_L \\ &= \frac{c_L}{2L} \end{aligned} \quad (4)$$

where c_L is the normalized lifetime of a master key K_L , i.e.,

$$c_L = T_L / A. \quad (5)$$

(2) Disclosure rate for SCHEME 2

Let t_i be the time when the i -th level key, K_i , is exposed and let T_i be the time when K_i is changed. Times t_i and T_i distributions are shown in Fig. 2. All t_i are considered as random variables.

The probability density function of disclosure of a key is

$$p(t) = \begin{cases} \frac{1}{A}; & 0 \leq t < A \\ 0; & \text{otherwise} \end{cases} \quad (6)$$

Disclosure rate ξ_L is given as

$$\begin{aligned} \xi_L = & \frac{1}{T_L} \{ E[I(T_1 - t_1)] \\ & + E[I(T_2 - t_2)] - E[I(T_1 - t_2)] \\ & + E[I(T_3 - t_3)] - E[I(T_2 - t_3)] \\ & + \dots \\ & + E[I(T_L - t_L)] - E[I(T_{L-1} - t_L)] \} \end{aligned} \quad (7)$$

where E shows expectation and I is the function defined as below.

$$I(t) = \begin{cases} t; & t \geq 0 \\ 0; & t < 0 \end{cases} \quad (8)$$

The minus terms in Eq. (7) come from the cases where the intervals (t_i, T_i) 's are crossed each other.

$E[I(T - t_i)]$ is calculated as,

$$E[I(T - t_i)] = \int_0^T p(t_1) dt_1 \int_{t_1}^T p(t_2 - t_1) dt_2 \dots \int_{t_{i-1}}^T p(t_i - t_{i-1})(T - t_i) dt_i$$

$$= \frac{1}{A^i} \int_0^{\min(T,A)} dt_1 \int_{t_1}^{\min(T,t_1+A)} dt_2 \dots \int_{t_{i-1}}^{\min(T,t_{i-1}+A)} (T-t_i) dt_i \quad (9)$$

As A is usually large, we assume that

$$T_i \leq A \quad (i = 1, 2, \dots, L). \quad (10)$$

Under this condition, $E[I(T-t_i)]$ is given as

$$\begin{aligned} E[I(T-t_i)] &= \frac{1}{A^i} \int_0^T dt_1 \int_{t_1}^T dt_2 \dots \int_{t_{i-1}}^T (T-t_i) dt_i \\ &= \frac{1}{A^i} \int_0^T dx_1 \int_0^{x_1} dx_2 \dots \int_0^{x_{i-1}} x_i dx_i, \quad (x_i = T-t_i) \\ &= \frac{T^{i+1}}{(i+1)! A^i} \end{aligned} \quad (11)$$

and ξ_L is given as,

$$\xi_L = \frac{1}{c_L} \sum_{i=1}^L \frac{c_i^{i+1} - c_{i-1}^{i+1}}{(i+1)!}, \quad (12)$$

where

$$c_i = T_i/A \quad (i=1, 2, \dots, L) \quad (13)$$

$$c_0 = 0 \quad (14)$$

Equations (4) and (12) relate disclosure rate ξ_L to the number of key levels L and the normalized keys lifetimes c_i . In SCHEME 2, it is desired that c_L be maximum, because K_L must be changed manually. It can be derived that c_L is maximized at

$$c_1 = c_2 = \dots = c_{L-1} = 0 \quad (15)$$

and the maximum c_L is given by

$$c_L = ((L+1) \cdot \xi_L)^{1/L}, \quad (16)$$

where ξ_L is considered as a parameter (See Appendix).

The $c_1 \dots, c_{L-1}$ influence on c_L will be investigated when $L=2$ in detail.

Figure 3 shows the relation between ξ_L and c_L , and Figure 4 shows the effectiveness of increase of L on the lifetime of key. We can see that SCHEME 2 is much stronger than SCHEME 1 for cryptanalysis.

3.2 Lifetimes of keys for the two level key cryptosystems

Lifetimes c_1 and c_2 in the two level key cryptosystems are investigated. These types of cryptosystems are fairly often used. From Eq.(12), the disclosure rate ξ_2 for SCHEME 2 is represented as

$$\xi_2 = \frac{c_1^2}{2c_2} + \frac{c_2^3 - c_1^3}{6c_2} \quad (17)$$

Figure 5 shows the relations between c_1 and c_2 for some values of a parameter ξ_2 .

Normalized lifetimes of keys c_2 and c_1 for SCHEME 2 are optimized by the rules below.

- 1) c_2 is maximized.
- 2) c_1 is maximized under the condition that c_2 nearly equals its maximum, i.e.,

$$c_2 = \sqrt{6 \cdot \xi_2} \quad (18)$$

Equation (18) comes from Eq.(16). In Figure 5, the maximum points of c_1 are given as the cross points on the graph where $c_2 = \sqrt{6 \cdot \xi_2}$. The line connecting these points is given by the equation,

$$\log c_1 - \log 10^{-1} = 3/2(\log c_2 - \log \sqrt{6 \times 10^{-1}}), \quad (19)$$

namely,

$$c_1 = 0.56 \cdot \varepsilon_2^{0.75}. \quad (20)$$

Equation (19) is given from Fig. 5 by rule of thumb.

The lifetimes of master keys T_2 are shown in Table 1, when DES is employed.

For example, when $\tau = 10^{-6}$ second, the master key for SCHEME 1 must be changed every year, though the master key for SCHEME 2 has only to be changed every 56 years to establish $\varepsilon \leq 10^{-4}$ (1 hour/year). When $\tau = 10^{-7}$ second, SCHEME 1 must change the master key every month, whereas SCHEME 2 has only to change the master key every 5 years. Therefore SCHEME 2 is superior to SCHEME 1 on the keys lifetime.

4. CONCLUDING REMARKS

The keys lifetimes necessary to attain a certain low disclosure rate have been investigated for two types of schemes. DES is employed as an encryption algorithm example. This paper employs the poorest attack, namely the exhaustive attack as a cryptanalysis. There may be a more effective attack. As results, we recommend to adopt SCHEME 2 and to change the master key 'at least' within a few years.

ACKNOWLEDGEMENT

The authors wish to thank Mr. Kato, Mr. Ishiguro and Mr. Goto of NEC Corporation for helpful suggestions.

REFERENCES

- [1] Lennon, R.E., "Cryptography Architecture for Information

- Security", IBM System J., vol.17, no.2, pp.138-150, 1978
- [2] Federal Information Processing Standards Publication No.46, National Bureau of Standards, 1977.
- [3] Okamoto, E. and Nakamura, K., "Key change Periods in Cryptographic Key Management Systems", The proceedings of the 7-th Symposium on Information Theory and Its Applications (in Japanese), pp.169-173, 1984.

APPENDIX

The aim here is to show that the maximum of c_L is given by Eq.(16) at $c_1=c_2=\dots=c_{L-1}=0$. From Eq. (12),

$$\frac{c_L^{L+1}}{(L+1)!} - \xi_L c_L = - \sum_{i=2}^L \frac{c_{i-1}^i}{i!} \left(1 - \frac{c_{i-1}}{i+1}\right) \quad (A1)$$

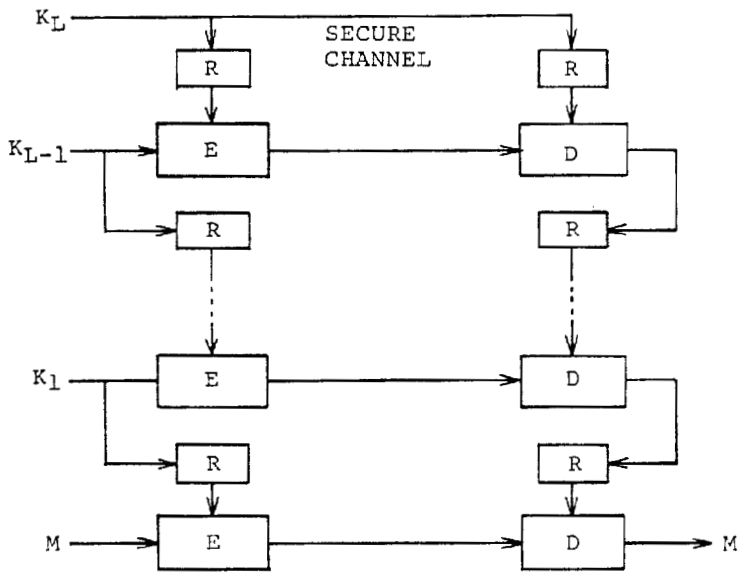
The right-hand side of Eq.(A1) is nonnegative from Eq.(10) and Eq.(13), while the left-hand side of Eq.(A1) is nonnegative if and only if

$$0 \leq c_L \leq ((L+1)! \cdot \xi_L)^{1/L} \quad (A2)$$

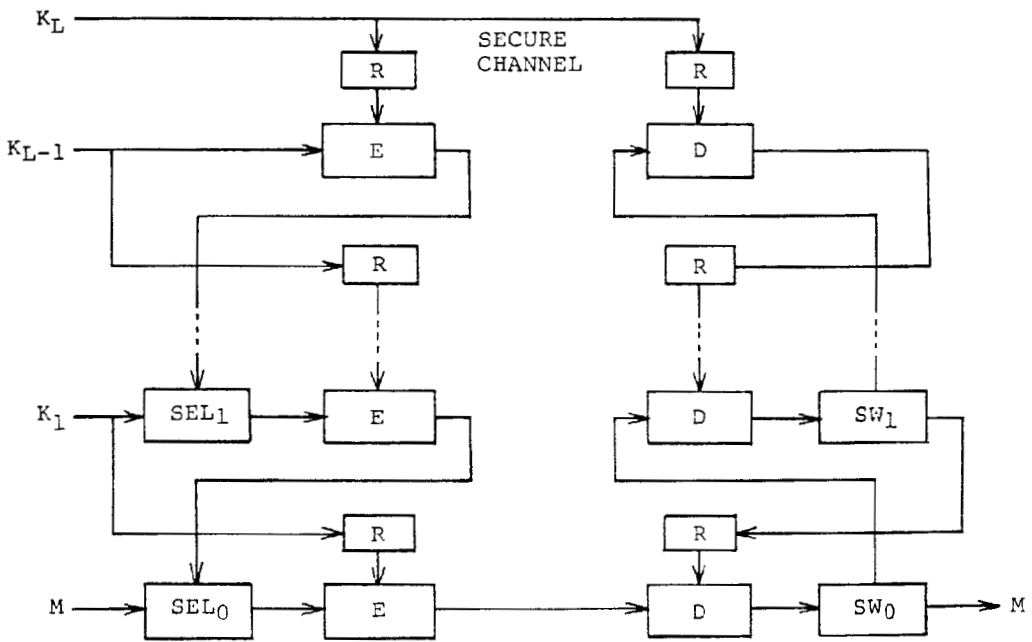
Hence the maximum of c_L is $((L+1)! \cdot \xi_L)^{1/L}$. When c_L is the maximum,

$$c_L = ((L+1)! \cdot \xi_L)^{1/L}, \quad (A3)$$

c_1, c_2, \dots, c_{L-1} are all zero, because the right-hand side of Eq.(A1) is zero and c_{i-1} does not equal $i+1$ from Eq.(10) and (13).

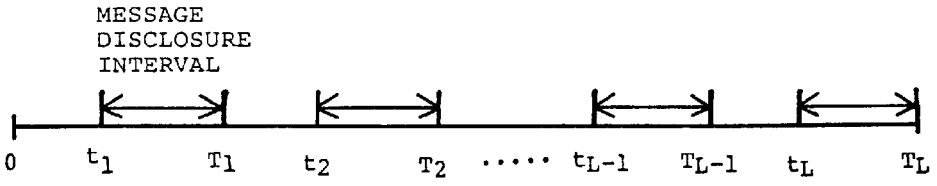


(a) SCHEME 1



(b) SCHEME 2

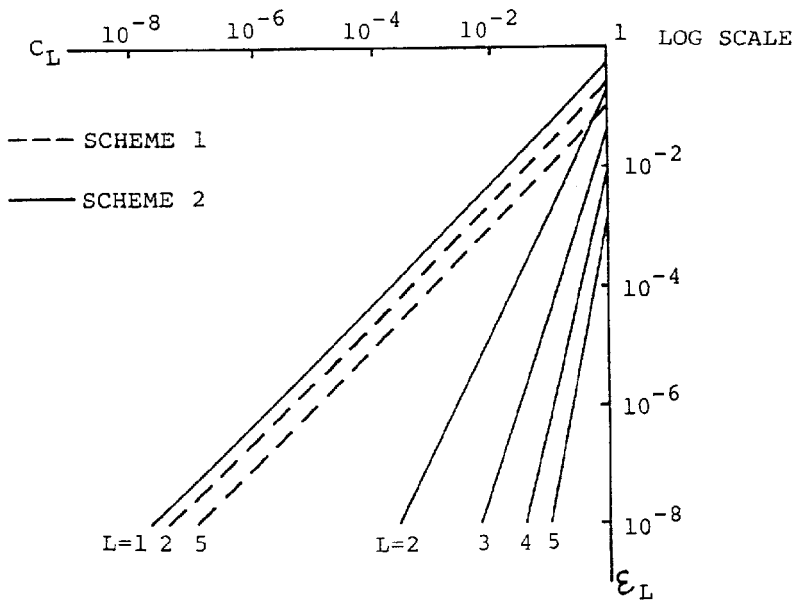
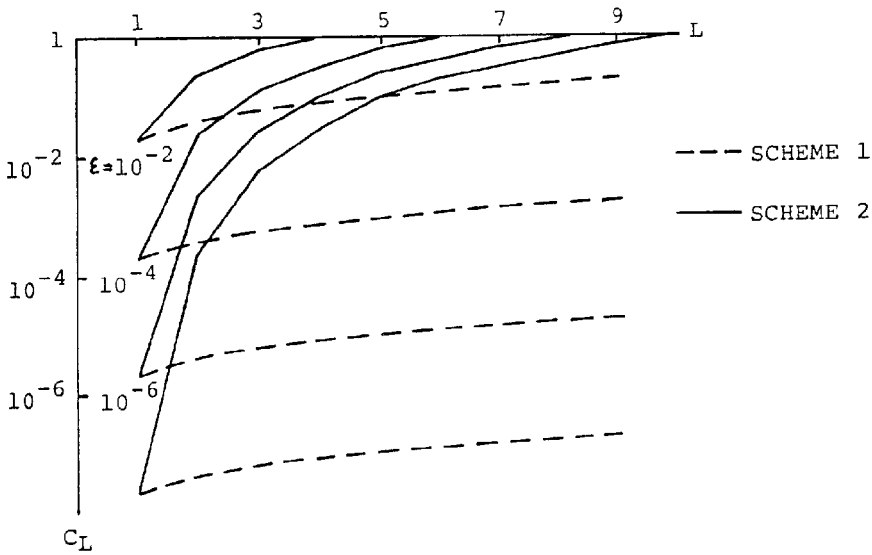
Figure 1. Key distribution schemes



t_i : time when K_i is found

T_i : time when K_i is changed

Figure 2. Times t_i and T_i distribution

Figure 3. Relation between C_L and ϵ_L Figure 4. Relation between C_L and L

$$\varepsilon_2 = \frac{c_1^2}{2c_2} + \frac{c_2^3 - c_1^3}{6c_2}$$

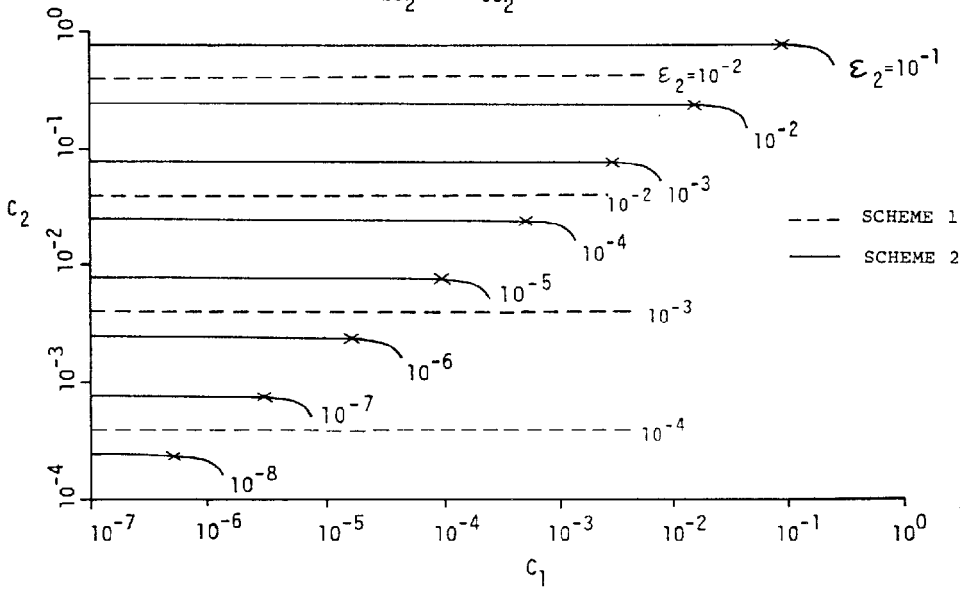


Figure 5. Relation between c_1 and c_2 with parameter ε_2

Table 1. Lifetimes of master keys for
2-level key cryptosystems

a) Lifetime of master key T_2 for SCHEME 1

Encrypting Time τ Disclosure Rate ϵ	$10^{-6}(s)$	$10^{-7}(s)$	$10^{-8}(s)$
10^{-1}	914 (y)	91.4 (y)	9.14 (y)
10^{-2}	91.4	9.14	334 (d)
10^{-3}	9.14	334 (d)	33.4
10^{-4}	334 (d)	33.4	3.34
10^{-5}	33.4	3.34	8.01 (h)
10^{-6}	3.34	8.01 (h)	48.0 (m)
10^{-7}	8.01(h)	48.0 (m)	4.80 (m)

b) Lifetime of master key T_2 for SCHEME 2

Encrypting Time τ Cryptanalysis Ratio ϵ	$10^{-6}(s)$	$10^{-7}(s)$	$10^{-8}(s)$
10^{-1}	1770 (y)	178 (y)	17.8 (y)
10^{-2}	560	56	5.6
10^{-3}	178	17.8	1.78
10^{-4}	56	5.6	204 (d)
10^{-5}	17.8	1.78	64
10^{-6}	5.6	204 (d)	20.4
10^{-7}	1.78	64	6.4
10^{-8}	204 (d)	20	2

Correlation Immunity and the Summation Generator

Rainer A. Rueppel

CMRR

University of California, San Diego

La Jolla, CA, 92093

Abstract:

It is known that for a memoryless mapping from $GF(2)^N$ into $GF(2)$ the nonlinear order of the mapping and its correlation-immunity form a linear tradeoff. In this paper it is shown that the same tradeoff does no longer hold when the function is allowed to have memory. Moreover, it is shown that integer addition, when viewed over $GF(2)$, defines an inherently nonlinear function with memory whose correlation-immunity is maximum. The summation generator which sums N binary sequences over the integers is shown as an application of integer addition in random sequence generation.

1. Introduction

Boolean functions from $GF(2)^n$ into $GF(2)$ are commonly found in cryptographic applications. Usually they are designed to be nonlinear and to produce a balanced output, and, often one finds the additional requirement that from knowledge of the output bit it should not be possible to reliably guess one or more input bits. Consider for example DES, where the S-boxes define nonlinear mappings from $GF(2)^4$, (or $GF(2)^6$ respectively), into $GF(2)$ chosen in such a way that little statistical dependency is created between the output bit and one or more input bits. Or consider a classical running-key generator for use in a stream cipher system. Such a running-key generator consists of N driving linear feedback shift registers (LFSRs) and some nonlinear function operating on the N output sequences in order to produce the running-key. Siegenthaler [1] has recently shown that several of the previously published running-key generators employed

nonlinear functions which created statistical dependencies between single input and output variables and therefore allowed 'divide-and-conquer' attacks using correlation techniques. These results stimulated some interest in functions which can resist the correlation attack. The concept of m -th order correlation-immunity for combining functions [2] was introduced as a measure of their resistance against such correlation attacks. (But correlation-immunity is not confined to running-key generators. In fact, if a boolean function is found to be m -th order correlation-immune, it means that there is no statistical dependency between the output variable and any subset of m input variables, provided the input variables are independent and uniformly distributed). Unfortunately, for such memoryless combining functions f there exists a tradeoff between the attainable nonlinear order and the attainable level of correlation-immunity [2]. If k and m denote the nonlinear order and the order of correlation-immunity of f , respectively, then

$$k + m \leq N-1 \quad \text{for } 1 \leq m \leq N-2. \quad (1)$$

Thus, the more correlation-immunity, the smaller the nonlinear order of f and consequently the smaller the linear complexity of the running-key, and vice versa. Moreover, functions which satisfy (1) with equality are difficult to find.

In section 2 we shall show that this inconvenient tradeoff can be avoided by proper use of memory in the nonlinear combining function. In fact, one bit of memory suffices to obtain nonlinear combiners that are maximally correlation-immune and have maximum nonlinear order at the same time. In section 3 we shall demonstrate that integer (or real) addition, which is an extremely nonlinear operation when considered over $GF(2)$, inherently defines a maximally correlation-immune combiner. Moreover, we will apply integer addition in random-sequence generation and give evidence that the resulting key stream is highly complex.

Throughout most of this paper, $GF(2)$ is taken as the underlying field of computation; therefore, unless otherwise stated, formulas are assumed to be computed over $GF(2)$.

2. Correlation-Immunity of Nonlinear Combiners

In order to investigate the statistical dependencies introduced by the nonlinear combiner itself (and not by the sources which feed it) we shall assume that the input sequences to the nonlinear combiner are sequences of independent and uniformly distributed binary random variables.

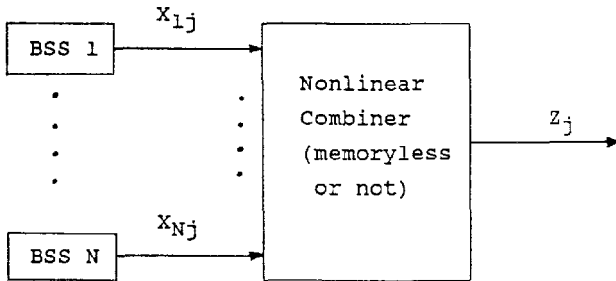


Fig. 1. Information-theoretic model used to define correlation-immunity. (BSS = Binary Symmetric Source)

Several authors ([2],[3]) investigated the correlation-immunity of nonlinear combiners, but always under the assumption that the combiner is memoryless. A memoryless nonlinear function is termed correlation-immune of order m [2] if the mutual information between the output variable and any subset of m input variables considered jointly is zero. For a memoryless combiner time is immaterial, since at any time the output only depends on the current input variables. Now allow the nonlinear combiner to contain memory which, in fact, converts it into a finite-state machine (FSM). Let S_0 denote the initial content of the combiner's memory and define $X_i^j = X_{i1}, X_{i2}, \dots, X_{ij}$, for $1 \leq i \leq N$. For any FSM we may write

$$z_j = F(X_1^j, \dots, X_N^j, S_0) \quad (2)$$

As a natural extension of the above definition of correlation-immunity for memoryless combiners we shall say that a nonlinear combiner with memory is correlation-immune of order m if the mutual information between the output sequence and any subset of m input sequences is zero, that is, if

$$I(Z^j; X_{i_1}^j, \dots, X_{i_m}^j) = 0 \quad j > 0 \quad (3)$$

$$1 \leq i_1 < i_2 < \dots < i_m \leq N$$

In this case the output sequence is statistically independent of any m input sequences considered jointly. In many cryptographic applications it is required that the output sequence should resemble as closely as possible a truly random sequence. For example, in a running-key generator, it must not be possible to reliably guess the next key bit regardless of how many prior key bits have been observed. In the information-theoretic model this corresponds to requiring that $\{Z_j\}$ forms a sequence of independent and uniformly distributed random variables. Under this constraint the definition (3) is equivalent to

$$I(Z_j; X_{i_1}^j, X_{i_2}^j, \dots, X_{i_m}^j, Z^{j-1}) = 0 \quad j > 0 \quad (4)$$

$$1 \leq i_1 < i_2 < \dots < i_m \leq N$$

Definition (4) is intuitively pleasing: knowing all prior output bits and knowing (or guessing) jointly any m input sequences does not provide any information whatsoever on the next output bit. To prove the equivalence of (3) and (4), let $m=1$ and decompose (3) in the following way,

$$I(Z^j; X^j) = I(Z^{j-1}; X^j) + I(Z_j; X^j | Z^{j-1}) = 0.$$

Mutual informations are always greater or equal to zero; hence it must hold

$$I(Z_j; X^j | Z^{j-1}) = H(Z_j | Z^{j-1}) - H(Z_j | X^j, Z^{j-1}) = 0.$$

Taking into account that $\{Z_j\}$ forms an i.i.d. sequence, we arrive at

$$I(Z_j; X^j | Z^{j-1}) = I(Z_j; X^j, Z^{j-1}) = 0$$

which establishes the equivalence. For an in-depth treatment of the different definitions of correlation-immunity we refer to [5]. Now let the function F of (2) have the form

$$Z_j = \sum_{i=1}^N X_{ij} + F'(X_1^{j-1}, \dots, X_N^{j-1}, S_0) \quad (5)$$

where the current input variables X_{1j}, \dots, X_{Nj} are summed and added to an arbitrary function F' of all previous input variables and of the initial state S_0 . Suppose we know the complete history of the nonlinear combiner F and all but one, say X_{ij} , of the current input variables. We then may rewrite (5) as

$$Z_j = X_{ij} + Y_j \quad (6)$$

where Y_j summarizes our knowledge about the device. The fact that X_{ij} is drawn independently of Y_j from a uniform distribution implies that Z_j and Y_j are statistically independent and that Z_j is also uniformly distributed. This can also be seen from the fact that (6) corresponds to sending Y_j through a memoryless binary symmetric channel with capacity 0, thereby ensuring that Z_j is uniformly distributed and statistically independent of Y_j . Hence, any nonlinear combiner of the form (5) is $(N-1)$ st-order correlation-immune, which is in fact the maximum order of immunity possible. Moreover, the function F' in (5) is not restricted in any way and may consequently be chosen to be of maximum nonlinear order. In particular, one memory cell suffices in order to realize a combiner with maximum correlation-immunity and with maximum nonlinear order. For this case the FSM equations may be written as

$$Z_j = \sum_{i=1}^N X_{ij} + S_{j-1} \quad (7a)$$

$$S_j = f(X_{1j-1}, \dots, X_{Nj-1}, S_{j-1}) \quad (7b)$$

Equations (7) describe an FSM with finite memory of 1 bit. If the next state is computed irrespectively of the previous state, then the FSM (7) is said to have a finite input memory of 1 bit (it can be realized with a pure feedforward structure).

At this point it is illustrative to consider a practical example. Pless [4] proposed in 1977 a running-key generator which contains as basic building block a 2-LFSR-subgenerator. In this subgenerator a J-K Flip-Flop acts as the nonlinear element which combines the 2 LFSR sequences. A J-K Flip-Flop defines a one-bit FSM whose state just contains the previous output bit and whose output and next-state functions therefore coincide. Its behavior is completely described by

$$Z_j = X_{1j} + Z_{j-1}(1 + X_{1j} + X_{2j}) \quad (8)$$

Considering X_{1j} , X_{2j} , and Z_{j-1} as the 3 input variables to a memoryless mapping f defined by (8) we may compute the Walsh transform $S_f(w)$ [3] of f . Fig. 2 displays the result

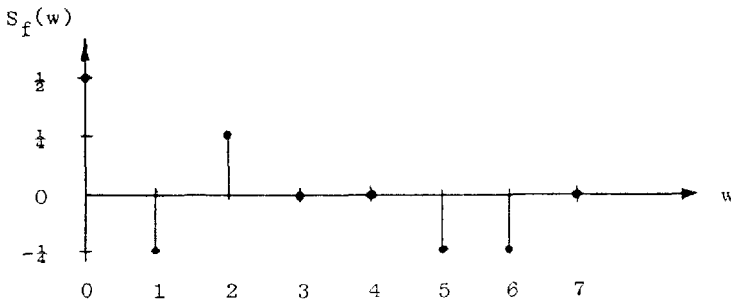


Fig. 2. Walsh transform of the boolean mapping defined by (8)

The graph of Fig. 2 may be interpreted as follows: let (w_0, w_1, w_2) denote the binary representation of w , where $0 \leq w \leq 7$. If the Walsh transform $S_f(w)$ is nonzero at some $w > 0$ then the mutual information $I(Z_j; w_0 X_{1j} + w_1 X_{2j} + w_2 Z_{j-1})$ is greater than zero. Moreover, the value of the Walsh transform at this w gives an exact account of how much statistical dependency is introduced. For instance, the peaks in the Walsh transform at $w = 1$ and $w = 2$ in Fig. 2 tell us that the output bit Z_j is neither independent of X_{1j} nor of X_{2j} . The value $-1/4$ at $S_f(1)$ tells us that the probability that Z_j coincides with X_{1j} is $3/4$. Equivalently, the value $+1/4$ at $S_f(2)$ tells us that the probability that Z_j coincides with X_{2j} is $1/4$. On the other hand, since $S_f(4)$ is zero Z_j is independent of Z_{j-1} . Consequently, if a J-K Flip-Flop is fed by two binary symmetric sources it will produce a sequence of independent and uniformly distributed binary random variables (as desired), but this output sequence will exhibit a strong correlation with either input sequence. Thus, the correlation-immunity of a J-K Flip-Flop is zero.

Comparing (2) and (4) we notice that maximum correlation-immunity of F was obtained by separating the N current input variables from an arbitrary function F' of only prior input variables. In general, any desired level m of correlation-immunity can be obtained by separating $m+1$ input variables each taken from a different input sequence and possibly with a different time index, but disallowing their use in the arbitrary function F' .

3. The Summation Principle

Let a and b be two integers, whose binary representation is given as $a = a_{n-1}2^{n-1} + \dots + a_1 2 + a_0$ and $b = b_{n-1}2^{n-1} + \dots + b_1 2 + b_0$, respectively. Let $z = a + b$ be the real sum of the two integers and assume that the sum is computed bit-serially in $GF(2)$ from the binary representations of a and b . Then we may write, with increasing nonlinear order of the binary functions producing the j -th bit

$$\begin{aligned}
 z_0 &= a_0 + b_0 \\
 z_1 &= a_1 + b_1 + a_0 b_0 \\
 z_2 &= a_2 + b_2 + a_1 b_1 + a_1 a_0 b_0 + b_1 a_0 b_0
 \end{aligned}
 \tag{9}$$

or, we may express z_j recursively for $0 \leq j \leq n$

$$z_j = f_1(a_j, b_j, c_{j-1}) = a_j + b_j + c_{j-1} \tag{10a}$$

$$c_j = f_2(a_j, b_j, c_{j-1}) = a_j b_j + (a_j + b_j) c_{j-1} \tag{10b}$$

where c_{j-1} represents the carry-bit from the less significant bits to bit j of the sum. Fig. 3 illustrates the principle.

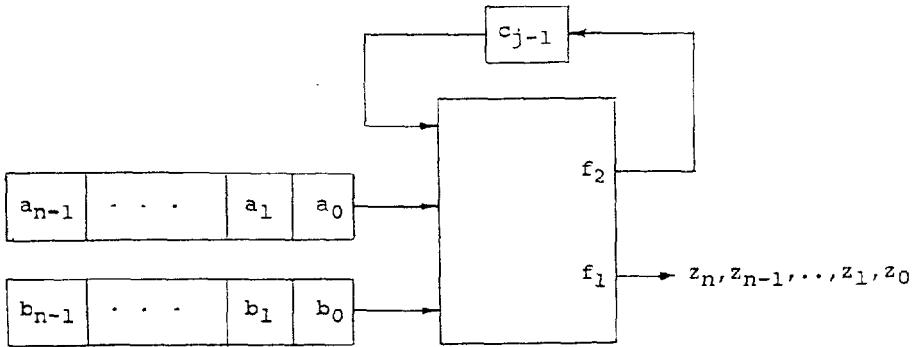


Fig. 3. Time-sharing of a 3-bit adder to produce bit-serially the real sum of two n -bit integers.

When the two input shift registers in Fig. 3 are initially loaded with the binary representation (least-significant bit first) of the two integers and when the feedback memory cell is initially zero, then after $(n+1)$ clock cycles the $(n+1)$ bits corresponding to the binary representation of the real sum will have appeared serially at the output. In fact, the real adder of Fig. 3 defines a finite-state machine with output and next-state functions according to (10), and, surprisingly enough, it directly realizes a correlation-immune combining function as defined in (7). Note that f_1 defines the GF(2)-sum of the input variables and thus accounts for the correlation immunity, while f_2 defines the GF(2)-sum of all second-order products of the input variables and thus implements a

memoryless nonlinear mapping. The memory-cell is used to hold the carry-bit from the $(j-1)$ -st to the j -th position of the sum and carries all the nonlinear influence of the less significant bits. These observations suggest that real addition could be useful in running-key generation. The simplest running-key generator based on this summation principle may be obtained by adding two (or in general N) infinite integers whose binary representations are periodic sequences generated by suitable LFSRs. We shall call any such generator a summation-generator. It is apparent from the linear form of the output function (10a) that whenever at least one input sequence consists of independent and uniformly distributed random variables so will also the output sequence. Besides statistical properties of a generator one is often interested in the period of a generator and its linear complexity (that is, the length of the shortest LFSR that is able to emulate the generator for a given output sequence).

Property 1:

Let $\{a_j\}$ and $\{b_j\}$ be two binary sequences with least periods T_1 and T_2 respectively. When $\{z_j\}$ denotes the real sum of $\{a_j\}$ and $\{b_j\}$, expressed in radix-2 form, and if $\gcd(T_1, T_2) = 1$, then $\{z_j\}$ has least period $T_1 T_2$.

Proof:

Define the rational fraction s associated to a sequence $\{s_j\}$ of period T as

$$s = \frac{\sum_{j=1}^T s_{T-j} 2^{-j}}{2^T - 1} = \frac{p}{q}$$

where $\gcd(p, q) = 1$. The period T may be found from q as the multiplicative order of 2 modulo q . Therefore we may write $a = p_1/q_1$ and $b = p_2/q_2$. The real sum of the sequences directly corresponds to the real sum of the rational fractions. Thus

$$a + b = \frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{p_1 q_2 + p_2 q_1}{q_1 q_2} = c + \frac{n}{q_1 q_2}$$

where we identify n/q_1q_2 as the rational fraction representing the real sum sequence $\{z_j\}$ and c , which is either 0 or 1, as the carry digit from one period of the sum sequence to the next. We note that $\gcd(n, q_1q_2)=1$ because $\gcd(q_1, q_2) = \gcd(p_1, q_1) = \gcd(p_2, q_2)=1$, and that $\gcd(2, q_1q_2)=1$ since q_1 divides $2^{T_1}-1$ and q_2 divides $2^{T_2}-1$. Then the period T of the real sum sequence $\{z_j\}$ is given by the multiplicative order of 2 modulo q_1q_2 . Since q_1 and q_2 are relatively prime it follows from the Chinese remainder theorem that T is equal to the product of the multiplicative orders of 2 modulo q_1 and 2 modulo q_2 . Hence $T=T_1T_2$.

Property 1 may easily be generalized to the sum of N periodic sequences in radix- r representation.

Now assume that the real adder is fed by two maximum-length sequences whose minimal polynomials have relatively prime degree L_1 and L_2 . This implies that their periods are relatively prime and thus, by property 1, the period T of the real sum sequence is $(2^{L_1}-1)(2^{L_2}-1)$ which value also provides an upper bound to the linear complexity of $\{z_j\}$. When the above two m -sequences are multiplied termwise then the resulting product sequence will have a minimal polynomial of degree L_1L_2 (i.e. linear complexity L_1L_2) all of whose roots are from $GF(2^{L_1L_2}) - GF(2^{L_1}) - GF(2^{L_2})$. The interesting question now is how the feedback memory of the real adder affects the linear complexity of the real sum sequence. From (9) we see that the order of the products involved in the direct description of the function producing z_j grows linearly with time. A finite-state machine is said to have finite input memory M if M is the least integer such that the output digit at time j may be expressed as a function of the input variables at times $j-M, \dots, j-1, j$. Clearly the FSM as described by (10) has in general infinite input memory. But whenever the input sequences to the real adder produce a pair of zeros or ones, then the state of the adder FSM is set to a value independent of the preceding states and input values. In particular, when periodic input sequences are used which produce at least a common pair of zeros or ones within the period of the output sequence (which is certainly true for the above pair of m -sequences), then the input memory M will be finite with respect to the particular driving sequences. This allows to convert the feedback structure of the nonlinear combiner (10) into a feedforward structure of input memory M (corresponding to a maximum

nonlinear order of $M+1$ in the functional description (9)). From the feedforward function it is then possible to calculate (or at least bound) the associated linear complexity of the output sequence. In fact, one may prove that real addition of binary sequences is so nonlinear that from the available L_1 elements in $GF(2^{L_1})$ and L_2 elements in $GF(2^{L_2})$, (which are the roots of the two primitive minimal polynomials), it may generate every element in $GF(2^{L_1 L_2}) - GF(2^{L_1}) - GF(2^{L_2})$.

Property 2:

Let $\{a_j\}$ and $\{b_j\}$ be two binary m -sequences whose primitive minimal polynomials have relatively prime degrees L_1 and L_2 . When $\{a_j\}$ and $\{b_j\}$ are added over the reals then the real sum sequence $\{z_j\}$ exhibits linear complexity LC close to its period length,

$$LC(\{z_j\}) \leq (2^{L_1}-1)(2^{L_2}-1) \quad (11)$$

with near equality.

Instead of giving the proof which is straightforward but rather tedious, we will display some simulation results [6] which confirm that the bound (11) is extremely tight. In fact, no serious degeneracy was ever found, which suggests that integer addition is an inherently good nonlinear function.

N	L_1	L_2	L_3	T	LC
2	3	4		105	≥ 100
	3	5		217	≥ 208
	4	5		465	≥ 455
3	2	3	5	651	≥ 641

Table 1. Small-scale simulations giving evidence that the bound (11) is very tight (for explanation of the table see text below).

In table 1, the column labeled N gives the number of m -sequences

added over the reals; the columns labeled L_1 , L_2 , and L_3 give the degrees of the minimal polynomials of the m-sequences which were added; the column labeled T gives the period of the sum sequences; the column labeled LC displays the smallest linear complexity obtained for all possible combinations of different primitive minimal polynomials of the mentioned degrees. For instance, the first row tells us that each of the different m-sequences of degree 3 (there are 2) was separately added to each of the different m-sequences of degree 4 (there are 2), and never was a linear complexity of smaller than 100 obtained.

Although it seems counterintuitive, integer (real) addition is extremely nonlinear when viewed over $GF(2)$. The results in this section show that given two integers whose binary representations have very low (linear) complexity then their real sum may have very high (linear) complexity. This of course depends whether use was made of the nonlinear potential of real addition. Suppose, for example, we add the two integers whose binary representations are the sequences 0101.. and 1010.. each having linear complexity 2. The result is the all-1 sequence of linear complexity 1. Note that in this case the real sum and the mod-2 sum of the 2 sequences are identical and, in fact, never a nonlinear contribution through a carry occurred.

Finally, we want to mention that a similar analysis applies to the 0/1-knapsack with N weights [7]. The i -th output bit of such a knapsack may be regarded as being produced by a boolean function from $GF(2)^N$ into $GF(2)$ whose coefficients are determined by the weights of the knapsack. One can prove that the nonlinear order of the function producing the i -th output bit is bounded from above by $\min(2^i, N)$. Therefore, roughly the $\log N$ least significant bits of a knapsack are considerably less nonlinear (are considerably weaker) than the remaining output bits.

References:

- [1] T. Siegenthaler, "Decrypting a Class of Stream Ciphers Using Ciphertext Only", IEEE Trans. on Computers, Vol. C-33, 1984.

- [2] T. Siegenthaler, "Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications", IEEE Trans. on Info. Th., Vol. IT-31, 1985.
- [3] Xiao Guo-zhen, J.L. Massey, "A Spectral Characterization of Correlation-Immune Combining Functions", submitted to IEEE Trans. on Info. Th.
- [4] V.S. Pless, "Encryption Schemes for Computer Confidentiality", IEEE Trans. on Computers, Vol. C-26, Nov. 1977.
- [5] T. Siegenthaler, "Design of Combiners to Prevent Divide and Conquer Attacks", Proceedings of Crypto 85, Santa Barbara, August 18-22, 1985.
- [6] U. Maurer, R. Viscardi, "Running-Key Generators with Memory in the Nonlinear Combining Function", Diploma Project, Swiss Federal Institute of Technology Zurich, Dec. 1984.
- [7] R.A. Rueppel, J.L. Massey, "The Knapsack as a Nonlinear Function", IEEE Symposium on Info. Th., Brighton, UK, June 24-28, 1985.

DESIGN OF COMBINERS TO PREVENT DIVIDE AND CONQUER ATTACKS

T. Siegenthaler
Institute for Communication Technology
Federal Institute of Technology
8092 Zurich, Switzerland

Abstract

A finite state machine driven by n independent sources each generating a q -ary sequence is investigated. The q -ary output sequence of that device is considered as the running-key sequence in a stream cipher. Possible definitions for Correlation-Immunity are discussed and a simple condition is given which ensures that divide-and-conquer attacks on such generators are prevented.

I Introduction

A common form of running key generators for use in stream ciphers consists of n driving sources and some combiner. We assume in this section that each of the sources independently generates a sequence of q -ary random variables and that a finite state machine (FSM) combines the n input sequences $x_{1,j}, x_{2,j}, \dots, x_{n,j}$ to an output sequence z_j , $j=0,1,\dots$. A FSM is a system with finite sets of input and output symbols, a finite set of states, a next-state function Ψ and an output function Φ :

$$\Psi: (\underline{x}_j, \underline{s}_j) \rightarrow \underline{s}_{j+1},$$

$$\Phi: (\underline{x}_j, \underline{s}_j) \rightarrow z_j,$$

where $\underline{x}_j = [x_{1,j}, x_{2,j}, \dots, x_{n,j}]$ and where $\underline{s}_j = [s_{1,j}, s_{2,j}, \dots, s_{k,j}]$ and $\underline{s}_{j+1} = [s_{1,j+1}, s_{2,j+1}, \dots, s_{k,j+1}]$ are the state vectors with the k q -ary components $s_{1,j}, s_{2,j}, \dots, s_{k,j}$ and $s_{1,j+1}, s_{2,j+1}, \dots, s_{k,j+1}$ at time instants j and $j+1$, respectively. \underline{s}_0 denotes the initial state. Fig. 1 shows a canonical representation for a FSM [1], driven by n q -ary sources.

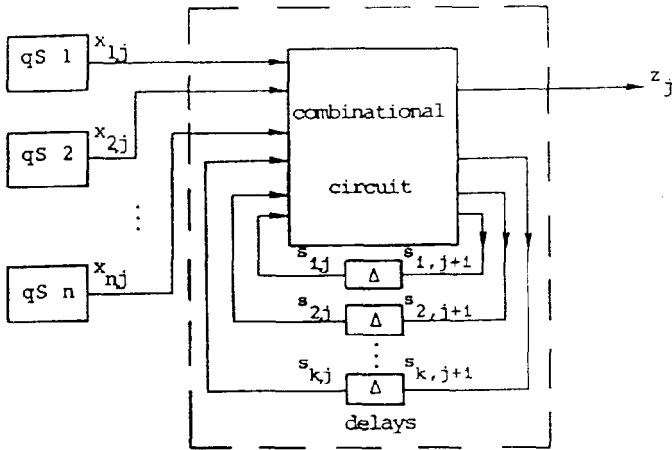


Fig. 1. A running key generator for use in stream ciphers.

A cryptanalyst possibly tries to break the above system by breaking the individual subkeys of the n sources. To prevent such divide and conquer attacks, the symbols generated by the FSM should be statistically independent on the symbols of one (or several) input sequences. In this note we give some results for FSM combiners.

II Correlation-Immunity of FSM combiners

A FSM combiner is called m -th order correlation-immune [2] if the mutual information between the running-key sequence z^j and every subset of m input sequences $x_{i1}^j, x_{i2}^j, \dots, x_{im}^j$, $1 \leq i_1 < i_2 < \dots < i_m \leq n$, is zero.

$$I(z^j; x_{i1}^j, x_{i2}^j, \dots, x_{im}^j) = 0, \text{ for all } j \geq 0, \quad (1)$$

where the superscript j means that all symbols up to time instant j are considered, e.g. $z^j = z_0, z_1, z_2, \dots, z_j$. Note that z^j contains $j+1$ symbols. The sequences $x_{i1}^j, x_{i2}^j, \dots, x_{im}^j$ are assumed to be independent of each other. Definitions (1) and (2) which is slightly stronger have been used by Rueppel [3,4].

$$I(z_j; z^{j-1}, x_{i1}^j, x_{i2}^j, \dots, x_{im}^j) = 0, \text{ for all } j > 0, \quad (2a)$$

and

$$I(z_j; x_{i1}^j, x_{i2}^j, \dots, x_{im}^j) = 0, \text{ for } j = 0. \quad (2b)$$

In this section it will be shown that (2) is too restrictive to be used as a definition in general but is useful for a special (but cryptographically significant) case. Moreover, an expression equivalent to that given in (1) is derived. For ease in notation we assume $m=1$, however, the result is easily extended to any m , $1 \leq m < n$. From (2a) we obtain

$$I(z_j; z^{j-1}, x_1^j) = I(z_j; z^{j-1}) + I(z_j; x_1^j | z^{j-1}) = 0, \quad j > 0. \quad (3)$$

Because mutual information is always positive, we must have

$$I(z_j; z^{j-1}) = H(z_j) - H(z_j | z^{j-1}) = 0, \quad j > 0, \quad (4)$$

and

$$I(z_j; x_1^j | z^{j-1}) = 0, \quad j > 0. \quad (5)$$

For stationary input sequences (4) means that an independence definition according to (2) implies an independent and identically distributed (i.i.d.) sequence z_0, z_1, \dots which, of course, isn't necessary for correlation-immunity. Fig. 2 gives an example for the restriction made with a definition according to (2). All variables are binary and we assume in this example that the input sequences are balanced and i.i.d.

Example 1:

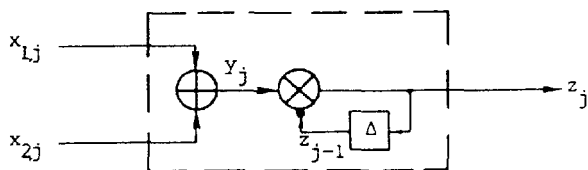


Fig. 2. A correlation-immune FSM with $I(z_j^j; x_1^j) = 0$ but $I(z_j; z^{j-1}, x_1^j) > 0$ for $i=1,2$. '•' denotes inversion.

We certainly have $I(x_i^j; y^j) = 0$ because the mod 2 addition at the input acts as a binary symmetric channel. From the data processing lemma follows that $I(x_i^j; z^j) \leq I(x_i^j; y^j) = 0$, for $i=1,2$. On the other hand, from $z_{j-1}=1$ follows that $z_j=0$, independently of the actual inputs. But this shows that $H(z_j | z^{j-1}) < H(z_j)$ or $I(z_j; z^{j-1}) > 0$ from which follows that $I(z_j; z^{j-1}, x_1^j) > 0$ for $i=1,2$.

Now we prove the following equality:

$$I(z^j; x_1^j) = \sum_{i=1}^j I(z_i; x_1^i | z^{i-1}), \quad j > 0. \quad (6)$$

First we have

$$I(x_1^j; z^j) = I(z^{j-1}; x_1^j) + I(z_j; x_1^j | z^{j-1}) \quad , j > 0. \quad (7)$$

The first term on the right hand side can be written as

$$\begin{aligned} I(z^{j-1}; x_1^j) &= H(x_1^j) - H(x_1^j | z^{j-1}) \quad , j > 0. \\ &= H(x_1^j) - H(x_1^{j-1} | z^{j-1}) - H(x_{1j} | x_1^{j-1} z^{j-1}) \quad , j > 0. \end{aligned} \quad (8)$$

From the independence of the input sequences and the additional assumption that the initial state s_0 is chosen independently of the input sequences, we have

$$\begin{aligned} H(x_{1j} | x_1^{j-1} z^{j-1}) &= H(x_{1j} | x_1^{j-1}, (x_1^{j-1}, \dots, x_n^{j-1}, s_0)) = \\ &= H(x_{1j} | x_1^{j-1}) \quad , j > 0, \end{aligned}$$

and therefore it follows that

$$I(z^{j-1}; x_1^j) = H(x_1^j) - H(x_1^{j-1} | z^{j-1}) - H(x_{1j} | x_1^{j-1}) \quad , j > 0.$$

The first term on the right hand side can be expanded as $H(x_{1j} | x_1^{j-1}) + H(x_1^{j-1})$ and therefore

$$I(z^{j-1}; x_1^j) = H(x_1^{j-1}) - H(x_1^{j-1} | z^{j-1}) = I(z^{j-1}; x_1^{j-1}) \quad , j > 0. \quad (9)$$

It follows from (7) and (9) that

$$I(x_1^j; z^j) = I(x_1^{j-1}; z^{j-1}) + I(z_j; x_1^j | z^{j-1}) \quad , j > 0. \quad (10)$$

(10) can be used iteratively to get (6). This completes the proof. From (6) immediately follows that the expressions given in (11) below are equivalent to expression (1) and therefore, are an equivalent definition for correlation-immunity of FSM's.

$$\begin{aligned} &I(z_j; x_{i1}^j, x_{i2}^j, \dots, x_{im}^j | z^{j-1}) = 0 \text{ for all } j > 0 \\ \text{and} \quad &I(z_j; x_{i1}^j, x_{i2}^j, \dots, x_{im}^j) = 0 \text{ for } j = 0, \end{aligned} \quad (11)$$

where m and i_1, i_2, \dots, i_m in (11) are defined as in (1). Note also that the independence definitions (1) and (2) are equivalent if and only if the FSM generates an i.i.d. output sequence.

III A design criterion for finite state machines

In practice it may often be difficult to work with expression (1). In this section we assume that the input sequences are independent of each other and i.i.d. and we work out a much simpler condition.

Theorem:

A sufficient condition for (1) to hold is that the current state \underline{s}_j and every set of m current inputs $x_{i1,j}, x_{i2,j}, \dots, x_{im,j}$, $1 \leq i_1 < i_2 < \dots < i_m \leq n$, are jointly statistically independent of the current output symbol z_j . If the FSM is a finite output memory machine which, moreover, generates an i.i.d. output sequence this condition is also necessary.

Note that it is sufficient due to the above theorem to fulfil some requirements on the memoryless output-function Φ independently on the chosen next-state function Ψ . To avoid unnecessary difficulties in notation the proof is given again for $m=1$ but is easily extended to any m , $0 < m < n$. First we have

$$H(z^j | x_1^j) = H(z_0 | x_1^j) + H(z_1 | z_0 x_1^j) + \dots + H(z_j | z^{j-1} x_1^j)$$

for a causal system with i.i.d. input sequences follows

$$H(z^j | x_1^j) = H(z_0 | x_1^0) + H(z_1 | z_0 x_1^1) + \dots + H(z_j | z^{j-1} x_1^j).$$

For the FSM of Fig. 1 we have

$$H(z^j | x_1^j) \geq H(z_0 | x_{1,0}) + H(z_1 | \underline{s}_1 x_{1,1}) + \dots + H(z_j | \underline{s}_j x_{1,j})$$

or

$$H(z^j | x_1^j) \geq H(z_0 | x_{1,0}) + \sum_{i=1}^j H(z_i | x_{1,i}, \underline{s}_i) \quad (12)$$

Note that for a finite output memory machine (where the state is identical to some finite number of output digits) equality holds in (12). Now we use

$$I(z^j; x_1^j) = H(z^j) - H(z^j | x_1^j)$$

and together with (12) we obtain

$$I(z^j; x_1^j) \leq H(z^j) - H(z_0 | x_{1,0}) - \sum_{i=1}^j H(z_i | x_{1,i}, \underline{s}_i).$$

The right hand side can be further increased by using

$$H(z^j) \leq \sum_{i=0}^j H(z_i) \quad (13)$$

and therefore

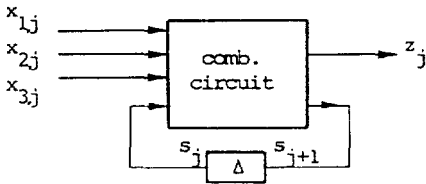
$$I(z^j; x_1^j) \leq H(z_0) - H(z_0 | x_{1,0}) + \sum_{i=1}^j [H(z_i) - H(z_i | x_{1,i}, \underline{s}_i)]$$

or

$$I(z^j; x_1^j) \leq I(z_0; x_{1,0}) + \sum_{i=1}^j I(z_i; x_{1,i}, \underline{s}_i), \quad (14)$$

where equality holds in (14) for a finite output memory machine which fulfills (13) with equality. The theorem follows immediately from the fact that $I(z_i; x_{1,i}, \underline{s}_i) = 0$ is equivalent to saying that the current input $x_{1,i}$ and the current state \underline{s}_i are jointly statistically independent of the current output z_i . Note that the FSM of Example 1 fulfills (1) even if the state and some inputs are not jointly statistically independent of the output. However, this is not a contradiction to the theorem because the finite output memory machine of Fig. 1 doesn't generate an i.i.d. sequence and therefore the necessary part of the theorem doesn't hold. The sequences in the following two examples have digits in $GF(2)$.

Example 2:



$$s_{j+1} = Y(x_{1j}, x_{2j}, x_{3j}, s_j)$$

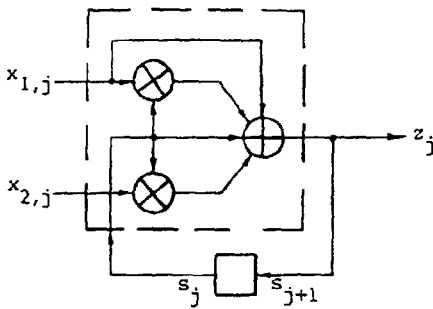
$$\Phi: z_j = x_{1j} \oplus x_{2j} \oplus s_j \cdot x_{2j} \oplus s_j \cdot x_{3j}$$

Fig. 3 A correlation-immune FSM with $n=3$, $m=1$.

The above FSM is correlation-immune with $m=1$ for any choice of Y due to the theorem of this section because x_{ij}, s_j are jointly statistically independent of z_j for $i = 1, 2, 3$. (For every choice of $x_{i,j}, s_j$ the output z_j is independently determined by the j -th digit of an i.i.d. sequence.)

Example 3:

The JK-FlipFlop (see Fig.4 for a logic equivalent) is an example for a finite output memory machine which generates an i.i.d. sequence when driven by two i.i.d. input sequences. However, it doesn't fulfil the necessary condition given in the theorem, as can be seen from the corresponding function Φ . For $s_j=0$ and any choice of $x_{1,j}$ we have $z_j = x_{1,j}$ and therefore s_j and $x_{1,j}$ are not jointly statistically independent of z_j .



$$\Phi: z_j = x_{1j} \otimes s_j \oplus x_{2j} \otimes s_j$$

$$\Psi: s_{j+1} = z_j$$

Fig. 4. A finite output memory machine which generates an i.i.d. output sequence but is not correlation-immune.

Conclusions

Definitions for correlation-immunity of general finite state machines have been discussed. The input sequences have been assumed to be independent of each other. It turned out that the definitions according to (1) and (11) are equivalent. The definition according to (2) is equivalent to that given in (1) if and only if the output sequence generated by the FSM is an i.i.d. sequence. Further, a simple sufficient condition for FSM's to be correlation-immune has been developed under the assumption that the input sequences are independent of each other and i.i.d. Moreover, it turned out that this condition is also necessary, if the FSM is a finite output memory machine which generates an i.i.d. output sequence.

Acknowledgment

The author is grateful to Dr. P. Schöbi from the Institute for Signal and Information Processing, Swiss Federal Institute of Technology, Zurich, for many helpful discussions and a thorough reading of the manuscript.

References

- [1] R.E. Miller, "Switching Theory", Vol. II, Sequential Circuits and Machines, John Wiley & Sons, New York, London, Sydney, 1965.
- [2] T. Siegenthaler, "Correlation-Immune Combining Functions for Cryptographic Applications", IEEE Tr. on Info. Theory, IT-30, No.5, Sept. 1984.
- [3] R. Rueppel, "New Approaches to Stream Ciphers", Thesis, Swiss Federal Institute of Technology, No. 7714, 1984.
- [4] ---, "How to Frustrate the Correlation Attack with one Bit of Memory" CRYPTO'85, Santa Barbara, Aug. 18-22, 1985.

On the Security of DES

Adi Shamir
Applied Mathematics
The Weizmann Institute
Rehovot, Israel
(abstract)

The purpose of this note is to describe some anomalies found in the structure of the S-boxes in the Data Encryption Standard. These anomalies are potentially dangerous, but so far they have not led to any successful cryptanalytic attack. While their significance is still unknown, they clearly demonstrate the deficiencies of current certification techniques and the need for provably secure cryptosystems.

Each S-box is a mapping from six input bits ABCDEF to four output bits WXYZ. Even though they are visually random, they have a lot of intentional structure, which seems to have a positive effect on the security of DES. However, the design criteria used by IBM and the NSA were never made public.

Fig. 1 describes our main observation. It circles all the WXYZ entries in which $W \oplus X \oplus Y \oplus Z = 0$ (0,3,5,6,9,10,12,15). There is a clear correlation between this function and input bit B (which determines the left/right half of each S-box). Furthermore, the minorities in each half are located in such a way that there are exceptionally simple boolean polynomials (XOR's of AND's) which describe the 64 values of $W \oplus X \oplus Y \oplus Z$ in each S-box with very small number of errors. A detailed description of these observations, along with possible lines of attack based on them, will appear in the full paper.

Remarks: (1) The correlation between the XOR of the outputs and input bit B was independently observed by Matthew Franklin from Berkeley in his M.Sc. Thesis (submitted May 1985). I am grateful to Gilles Brassard for bringing this to my attention.

(2) Preliminary analysis by Ernie Brickell and Don Coppersmith suggests that the observed properties of the S-boxes could be an unintentional consequence of some of the design criteria.

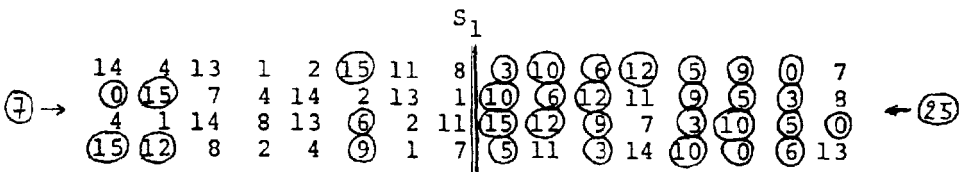


Figure 1
(continued on next page)

S_2

$\textcircled{10} \rightarrow$	$\textcircled{15}$	1	8	14	$\textcircled{6}$	11	$\textcircled{3}$	4	$\textcircled{9}$	7	2	13	$\textcircled{12}$	0	$\textcircled{5}$	$\textcircled{10}$		$\textcircled{22} \leftarrow$
	$\textcircled{3}$	13	4	7	$\textcircled{15}$	2	8	14	$\textcircled{12}$	0	1	$\textcircled{10}$	$\textcircled{6}$	9	11	$\textcircled{5}$		
	0	14	7	11	$\textcircled{10}$	4	13	1	$\textcircled{5}$	8	$\textcircled{12}$	$\textcircled{6}$	9	3	2	$\textcircled{15}$		
	13	8	$\textcircled{10}$	1	$\textcircled{3}$	$\textcircled{15}$	4	2	11	$\textcircled{6}$	7	$\textcircled{12}$	0	$\textcircled{5}$	14	$\textcircled{9}$		

 S_3

$\textcircled{21} \rightarrow$	$\textcircled{10}$	0	$\textcircled{9}$	14	$\textcircled{6}$	$\textcircled{3}$	$\textcircled{15}$	$\textcircled{5}$	1	13	$\textcircled{12}$	7	11	4	2	8		$\textcircled{11} \leftarrow$
	13	7	0	$\textcircled{9}$	$\textcircled{3}$	4	$\textcircled{6}$	$\textcircled{10}$	2	8	$\textcircled{5}$	14	$\textcircled{12}$	11	$\textcircled{15}$	1		
	13	$\textcircled{6}$	4	$\textcircled{9}$	8	$\textcircled{15}$	$\textcircled{3}$	0	11	1	2	$\textcircled{12}$	$\textcircled{5}$	$\textcircled{10}$	14	7		
	1	$\textcircled{10}$	13	0	$\textcircled{6}$	$\textcircled{9}$	8	7	4	$\textcircled{15}$	14	$\textcircled{3}$	11	$\textcircled{5}$	2	$\textcircled{12}$		

 S_4

$\textcircled{20} \rightarrow$	7	13	14	$\textcircled{3}$	0	$\textcircled{6}$	$\textcircled{9}$	$\textcircled{10}$	1	2	8	$\textcircled{5}$	11	$\textcircled{12}$	4	$\textcircled{15}$		$\textcircled{12} \leftarrow$
	13	8	11	$\textcircled{5}$	$\textcircled{6}$	$\textcircled{15}$	0	$\textcircled{3}$	4	7	2	$\textcircled{12}$	1	$\textcircled{10}$	14	$\textcircled{9}$		
	$\textcircled{10}$	$\textcircled{6}$	$\textcircled{9}$	0	$\textcircled{12}$	11	7	13	$\textcircled{15}$	1	$\textcircled{3}$	14	$\textcircled{5}$	2	8	4		
	$\textcircled{3}$	$\textcircled{15}$	0	$\textcircled{6}$	$\textcircled{10}$	1	13	8	$\textcircled{9}$	4	$\textcircled{5}$	11	$\textcircled{12}$	7	2	14		

 S_5

$\textcircled{6} \rightarrow$	2	$\textcircled{12}$	4	1	7	$\textcircled{10}$	11	$\textcircled{6}$	8	$\textcircled{5}$	$\textcircled{3}$	$\textcircled{15}$	13	0	14	$\textcircled{9}$		$\textcircled{26} \leftarrow$
	14	11	2	$\textcircled{12}$	4	7	13	1	$\textcircled{5}$	0	$\textcircled{15}$	$\textcircled{10}$	$\textcircled{3}$	$\textcircled{9}$	8	$\textcircled{6}$		
	4	2	1	11	$\textcircled{10}$	13	7	8	$\textcircled{15}$	$\textcircled{9}$	$\textcircled{12}$	$\textcircled{5}$	$\textcircled{6}$	$\textcircled{3}$	0	14		
	11	8	$\textcircled{12}$	7	1	14	2	13	$\textcircled{6}$	$\textcircled{15}$	0	$\textcircled{9}$	$\textcircled{10}$	4	$\textcircled{5}$	$\textcircled{3}$		

 S_6

$\textcircled{21} \rightarrow$	$\textcircled{12}$	1	$\textcircled{10}$	$\textcircled{15}$	$\textcircled{9}$	2	$\textcircled{6}$	8	0	13	$\textcircled{3}$	4	14	7	$\textcircled{5}$	11		$\textcircled{11} \leftarrow$
	$\textcircled{10}$	$\textcircled{15}$	4	2	7	$\textcircled{12}$	$\textcircled{9}$	$\textcircled{5}$	$\textcircled{6}$	1	13	14	0	11	$\textcircled{3}$	8		
	$\textcircled{9}$	14	$\textcircled{15}$	$\textcircled{5}$	2	8	$\textcircled{12}$	$\textcircled{3}$	7	0	4	$\textcircled{10}$	1	13	11	$\textcircled{6}$		
	4	$\textcircled{3}$	2	$\textcircled{12}$	$\textcircled{9}$	$\textcircled{5}$	$\textcircled{15}$	$\textcircled{10}$	11	14	1	7	$\textcircled{6}$	0	8	13		

 S_7

$\textcircled{9} \rightarrow$	4	11	2	14	$\textcircled{15}$	0	8	13	$\textcircled{3}$	$\textcircled{12}$	$\textcircled{9}$	7	$\textcircled{5}$	$\textcircled{10}$	$\textcircled{6}$	1		$\textcircled{23} \leftarrow$
	13	0	11	7	4	$\textcircled{9}$	1	$\textcircled{10}$	14	$\textcircled{3}$	$\textcircled{5}$	$\textcircled{12}$	2	$\textcircled{15}$	8	$\textcircled{6}$		
	1	4	11	13	$\textcircled{12}$	$\textcircled{3}$	7	14	$\textcircled{10}$	$\textcircled{15}$	$\textcircled{6}$	8	0	$\textcircled{5}$	$\textcircled{9}$	2		
	$\textcircled{6}$	11	13	8	1	4	$\textcircled{10}$	7	$\textcircled{9}$	$\textcircled{5}$	0	$\textcircled{15}$	14	2	$\textcircled{3}$	$\textcircled{12}$		

 S_8

$\textcircled{8} \rightarrow$	13	2	8	4	$\textcircled{6}$	$\textcircled{15}$	11	1	$\textcircled{10}$	$\textcircled{9}$	$\textcircled{3}$	14	$\textcircled{5}$	0	$\textcircled{12}$	7		$\textcircled{24} \leftarrow$
	1	$\textcircled{15}$	13	8	$\textcircled{10}$	$\textcircled{3}$	7	4	$\textcircled{12}$	$\textcircled{5}$	$\textcircled{6}$	11	0	14	$\textcircled{9}$	2		
	7	11	4	1	$\textcircled{9}$	$\textcircled{12}$	14	2	0	$\textcircled{6}$	$\textcircled{10}$	13	$\textcircled{15}$	$\textcircled{3}$	$\textcircled{5}$	8		
	2	1	14	7	4	$\textcircled{10}$	8	13	$\textcircled{15}$	$\textcircled{12}$	$\textcircled{9}$	0	$\textcircled{3}$	$\textcircled{5}$	$\textcircled{6}$	11		

Information theory without the finiteness assumption, II. Unfolding the DES

G. R. Blakley

Department of Mathematics
Texas A&M University
College Station, Texas 77843-3368

AMS(MOS) Subject Classifications:

03D15, 08A99, 15A99, 20B99, 20D99, 68B99, 68C99, 94A99

ACM CR Categories and Subject Descriptors:

E.3, E.4, F.2.0, F.2.1, G.1.3, J.7

Key Words

alphabet, arithmetic, associativity, Caesar cipher, code, codomain, commutativity, composite, confusion, continuous, cryptosystem, cyclic group, DES, diffusion, discrete, distributivity, domain, field, function, galois field, group, matrix, message, polyalphabet, position, product, ramp scheme, relation, replacement, ring, substitution, sum, symbol, symmetric group, threshold scheme, toroidal matrix, transposition, universal algebra, vector space.

Abstract

The DES is described in purely mathematical terms by means of confusion, diffusion and arithmetic involving a group of messages and a group of keys. It turns out to be a diffusion/arithmetic cryptosystem in which confusion plays no role, although the *S*-boxes effect an arithmetic operation of replacement (which is sometimes mistaken for confusion) as an important part of the encryption process.

1. Introduction

Group-theoretic structures appear to underly all of cryptography and error control. In particular, cryptosystems all appear to employ four groups: a group K of keys; a group A , called the alphabet, of symbols; a group P of positions which symbols can occupy; and a group A^P of messages, i.e. functions from P to A . Every cryptosystem is a pair (c, d) of self-maps of $K \times A^P$ and is thus, from a mathematical viewpoint, a pair of very large matrices c and d . The coding map c turns an encrypt key $k \in K$ and a plaintext message $m \in A^P$ into a decrypt key $\bar{k} \in K$ and a ciphertext message $\bar{m} \in A^P$. The decoding map d takes the pair (\bar{k}, \bar{m}) as inputs and recovers (k, m) . The keys k and \bar{k} are merely inverses of each other in the group K . In a conventional cryptosystem the group K is widely known and it is easy to produce the inverse \bar{k} of k . Not so in a public key cryptosystem. In either type of cryptosystem the ciphertext message \bar{m} depends in a complicated way on both k and m .

Interestingly, all cryptosystems appear to be built up on the basis of just three primitives:

(Shannon) confusion, a generalization of cryptographic substitution;

(Shannon) diffusion, a generalization of cryptographic transposition; and

arithmetic (in the sense of universal algebra operations derived from the composition laws associated with the groups K , A , P and A^P). One extremely important arithmetic operation is replacement, a generalization of the notion of a cryptographic codebook.

These notions of confusion, diffusion and arithmetic can now be precisely defined, and so the general definition of cryptosystem herein is at once less general and more abstract than the one [DI79, p. 398; KO81, p. 28; DE82, p. 7; BE82, pp. 125-130; ME82, p. 14-53] which appears in the literature to date.

The DES exhibits rich structure, and is therefore a good exemplar of this approach to cryptography. The four groups in question are as follows. The alphabet group A is the field $A = GF(2) = \mathbb{Z}/2\mathbb{Z}$ with two elements. The group P of positions is the ring $P = \mathbb{Z}/64\mathbb{Z}$ of integers modulo 64. Hence the group A^P of messages is the 64-dimensional vector space $A^P = (\mathbb{Z}/2\mathbb{Z})^{(\mathbb{Z}/64\mathbb{Z})}$ of 64-bit words. The key group K is a 56-dimensional vector subspace of A^P . When DES is expressed in these terms it becomes clear that it uses no confusion at all, merely diffusion and arithmetic. However, part of the arithmetic is a unary operation based on the S -boxes. Unary operations, replacements in our terminology, are reminiscent of confusions and are often mistaken for them.

2. Messages, codes, cryptosystems, confusion, diffusion, arithmetic

This paper continues and refines the approach begun in [BL83; BL85b]. The idea is to reformulate information-theoretic objects such as codes (both error-control codes and cryptographic codes) ciphers, cryptosystems, and ramp schemes [BL85a] in terms of group theory. By this means we hope to produce many new objects (both continuous [BL87] and discrete) of the sorts described above, as well as to gain a deeper understanding of the existing ones.

As far as cryptography goes, the idea is to define a message as a map $m : P \rightarrow A$ from a group P of symbol positions to a group A of alphabetic characters (i.e. symbols). A map between groups might be expected to be a group homomorphism. If the groups are topological groups it might be expected to be continuous. But cryptosystem designers often try to avoid “nice” algebraic, analytic or probabilistic structure. Even if messages (i.e. members of A^P) have significant algebraic, analytic or probabilistic structure, cryptosystems are often built so as to have as little such structure as possible. The set A^P is a group in a natural manner induced by the group structure on A . Composition of maps is indicated by the \circ operation symbol everywhere below. Thus $d \circ c$ is the map d following the map c , and $d * c$ is the product of d and c if a natural product operation $*$ exists.

Definition 2.1: Let K , A and P be groups. We call A the alphabet. We call P the group of symbol positions. We call A^P the group of messages. We call K the group of keys. A cryptosystem on A^P with keyspace K is a pair of maps

$$c : K \times A^P \rightarrow K \times A^P$$

$$d : K \times A^P \rightarrow K \times A^P$$

such that

$$(d \circ c)((k, m)) = d(c((k, m))) = (k, m)$$

for all $(k, m) \in K \times A^P$.

If we write

$$c((k, m)) = (\bar{k}, \bar{m})$$

it seems usually to be true that \bar{k} does not depend on m , but is merely the inverse of k in whatever arithmetic is natural on K . In DES we have

$$\bar{k} = -k = k$$

in a vector space K over $GF(2)$, whence $-k = k$. In RSA we have [BL85b, p. 332]

$$\bar{k} \equiv k^{-1} \bmod \lambda(p * q)$$

in a ring $Z/\lambda(p * q)Z$ in which k is invertible. In a simple substitution cipher the decode key \bar{k} is the permutation inverse k^{-1} of the encode key $k \in \text{SYM}(A)$. Here, as in [KO81, p. 65], we use the notation $\text{SYM}(A)$ for the symmetric group on the set A , i.e.

the group of all permutations of A . In a transposition cipher we similarly have $\bar{k} = k^{-1} \in \text{SYM}(P)$. The cryptext message \bar{m} , on the other hand, seems always to depend on both k and m . In fact cryptosystem designers often have to force some mutual compatibility on the group structures of A^P and K in order to make this dependence easy to calculate.

Definition 2.1 can certainly be generalized. We have assumed that the set A^P of plaintext messages is the same as the set of cryptext messages. This is often true, but doesn't have to be.

In short, a cryptosystem is a pair of matrices whose entries are chosen from the set of their (common) indices. This matrix structure does not necessarily make a cryptosystem easy to reconstruct or cryptanalyze. DES, for example, can be viewed as a 2^{56} by 2^{64} matrix with entries chosen from $GF(2)^{56} \times GF(2)^{64}$. Sometimes it is preferable to regard DES as a 2^{64} by 2^{64} matrix with entries chosen from $GF(2)^{64} \times GF(2)^{64}$, as we shall see in Section 3 below. An RSA is typically a $\phi(\lambda(p * q))$ by $p * q$ matrix with entries chosen from

$$K \times A^P = [Z/\phi(\lambda(p * q))Z] \times [Z/p * qZ],$$

where the primes p and q exceed 2^{250} .

Our thesis is that all known cryptosystems are built using only three notions: confusion, diffusion, and arithmetic. Confusion (a

generalization of substitution) is a selfmap

$$s : A \rightarrow A$$

of A or even merely a binary relation s on A . In other words a confusion acting on a message $m : P \rightarrow A$ is a member s of the power [HA60, p. 100] set $2^{A \times A}$. But often a confusion is a member s of A^A . There is a well known canonical injection

$$i : A^A \rightarrow 2^{A \times A}.$$

So the $s \in A^A$ definition is just a (most commonly encountered) case of the $s \in 2^{A \times A}$ definition. Actually we are sometimes driven even further than this (e.g. when we have to describe [BL85b, pp. 322-326] polyalphabetic substitutions [DE82, pp. 73-87] and one-time pads [DE82, pp. 86-87]). So our final definition of confusion is a family s of members of A^A , or even of members of $2^{A \times A}$. In ultimate generality, then, we have

Definition 2.2: Let A and P be groups. We call A^P the group of messages. A confusion on A^P is a family

$$s : I \rightarrow 2^{A \times A}$$

of binary relations on A . In particular, a family

$$s : I \rightarrow A^A$$

of self-maps of A is a confusion on A^P . Here, I is any index set. If I is a singleton and

$$s : I \rightarrow \text{SYM}(A)$$

then s is a monalphabetic substitution on A^P .

Here, as above, $\text{SYM}(A)$ is the group of permutations of A . Clearly

$$\text{SYM}(A) \subseteq A^A \subseteq 2^{A \times A}.$$

Similarly

$$\text{SYM}(P) \subseteq P^P \subseteq 2^{P \times P}.$$

Thus, by analogy with the definition of confusion, we have

Definition 2.3: Let A and P be groups. We call A^P the group of messages. A diffusion on A^P is a family

$$t : J \rightarrow 2^{P \times P}$$

of binary relations on P . In particular, a family

$$t : J \rightarrow P^P$$

of self maps of P is a diffusion on P . Here J is any index set. If J is a singleton and

$$t : J \rightarrow \text{SYM}(P)$$

then t is a transposition on A^P (or, at worst, an anagram on A^P).

This time the idea is that a diffusion acting on A^P is a selfmap

$$t : P \rightarrow P$$

of P or, at worst, a family of binary relations on P . As before, we allow the possibility of an entire family of self-maps of P , or even of an entire family of binary relations on P . Even such an object is called a diffusion.

The word *arithmetic* is taken in the sense of universal algebra [GR68]. Nullary, unary, binary, ternary, ..., qary, ... operations on the alphabet A (i.e. the set of “symbols” used) are arithmetic. So are such operations on the group P of symbol positions, on the group K of keys, on the group A^P of messages, or on the group $K \times A^P$. A particularly important type of arithmetic is a unary operation on A^P , i.e. a map

$$r : A^P \rightarrow A^P.$$

Definition 2.4: Let A and P be groups. We call A^P the group of messages. A replacement on A^P is a unary operation on A^P , i.e. a map

$$r : A^P \rightarrow A^P.$$

Definition 2.5: Let G be a group. The following objects are arith-

metric on G :

nullary operations $\hat{n} : \{\phi\} \rightarrow G$

unary operations $\hat{u} : G \rightarrow G$

binary operations $\hat{b} : G \times G \rightarrow G$

ternary operations $\hat{y} : G \times G \times G \rightarrow G$

\vdots

qary operations $\hat{q} : G \times G \times \dots \times G \rightarrow G$.

In this way we have defined arithmetic on the following structures related to a cryptosystem:

the group P of symbol positions;

the group A of symbols (the alphabet);

the group K of keys;

the group A^P of messages;

the group $K \times A^P$.

Usually arithmetic on A^P is induced by arithmetic on A , or arithmetic on P , or both. For example, if $b, c \in A^P$ then we have

$$b : P \rightarrow A$$

$$c : P \rightarrow A .$$

Let $\nabla : A \times A \rightarrow A$ be a binary operation on A . Then ∇ induces a natural binary operation (which, by the usual abuse of notation,

we will also call ∇) on A^P . We define $\nabla : A^P \times A^P \rightarrow A^P$ by subjecting

$$b\nabla c : P \rightarrow A$$

to the requirement that

$$(b\nabla c)(p) = b(p)\nabla c(p)$$

for every $p \in P$. If A is a field then A^P is a vector space over A . Its dimensionality is the cardinality of P .

Since a replacement is a unary operation on A^P , it follows that the notion of replacement is logically superfluous, being a special case of arithmetic on A^P . But we will nevertheless use the "replacement" terminology because this particular special case arises so often, and corresponds to the classical cryptographic notion of codebook.

There are a lot of groups K, A, P . So there are a lot of matrices

$$c : K \times A^P \rightarrow K \times A^P$$

The thesis this paper presents is to the effect that people who build cryptosystems always gravitate toward those matrices c which arise simply and naturally out of just confusion on A^P , diffusion on A^P , and arithmetic on A , on P , and on A^P . This often means they must forcibly relate K to A , or even to A and P in some, not always natural, manner.

An analogy to the thesis we present might be Cayley's theorem: If you want to understand groups, it suffices to understand permutations. There is probably no "Cayley theorem" to the effect that, if you want to understand cryptosystems, it suffices to look at confusion/diffusion/arithmetic cryptosystems. But our "Cayley thesis" (to the effect that people have never departed from the confusion/diffusion/arithmetic methodology so far in building cryptosystems) can have uses. If it is false, what is a historical counterexample? If it is true, why do people tend to do this? Either way, it is now possible to produce numerous useful cryptosystems using the confusion/diffusion/arithmetic methodology. It should be possible to exploit it to produce a taxonomy of cryptosystems. Will such a taxonomy be useful to cryptanalysts? To cryptosystem designers? Can we produce novel useful cryptosystems which are not confusion/diffusion/arithmetic cryptosystems?

3. An overview of DES as a confusion/diffusion/arithmetic cryptosystem

The highly structured DES is a good example of how the confusion/ diffusion/ arithmetic approach to cryptosystem structure works. Recall that arithmetic includes replacement (a unary operation on the message group A^P). It also includes constants (nullary operations) and binary operations on the collection K of keys, on the domain P of the collection of messages, on the codomain A of

the collection of messages, on the collection A^P of messages itself (though this last is usually induced by a related operation on the codomain A), and on the cartesian product $K \times A^P$ of the key collection with the message collection.

The standard descriptions [BE82, pp. 267-285; DE82, pp. 91-97; KO81, pp. 240-249; ME82, pp. 141-165] of DES describe its underlying structure in a hybrid terminology which mixes mathematical, mechanical and electrical metaphors. Moreover, though the descriptions in [BE82; DE82; KO81; ME82] are logically equivalent, they are not the same in detail. In particular it is commonplace to index rows and columns of S -boxes by the set $Z/16Z = \{0, 1, 2, \dots, 14, 15\}$. But Konheim goes on to use 0 as the index of the first element of every set he encounters, whereas Denning often uses 1 as the index of the first member of a set. We invariably follow Konheim's [KO81] usage herein.

Our description will be written in a top-down fashion. This section will give a brief unmotivated overview of how to describe DES in confusion/diffusion/arithmetic terms. Sections 4-9 will then go into the details. Our indebtedness to [DA84] should become obvious. We start by defining the notion of toroidal matrix. A matrix over a ring R is, of course, a function

$$M : B \times C \rightarrow R$$

whose domain is a cartesian product, $B \times C$ and whose codomain is

the ring R . If both B and C are cyclic groups one thinks intuitively geometrically of the matrix M as an array of numbers written on a bagel, rather than as a bunch of numbers written in a rectangle. This attitude is very natural and helpful in following our description of DES below. Consequently we will often use the phrase “toroidal matrix” to direct the reader’s attention to the fact that the cartesian factors B and C of the index set $B \times C$ of M are both cyclic groups whose cyclic structure is explicitly or implicitly used in constructing or manipulating M .

We will adopt the abbreviation

$$\Delta = (Z/2Z)^{[(Z/96Z) \times (Z/2Z)]}$$

for the vector space of all 96 by 2 toroidal matrices with entries belonging to the field $Z/2Z$, as well as the abbreviation

$$D = (Z/96Z) \times (Z/2Z)$$

for the index set of these matrices. Thus we have

$$\Delta = (Z/2Z)^D.$$

The description of DES starts with a plaintext message block

$$\overline{m} : Z/64Z \rightarrow Z/2Z$$

i.e. a 64-bit word, and a key

$$\overline{k} : Z/64Z \rightarrow Z/2Z$$

i.e. another 64-bit word. This latter word is formed in such a manner [DE82, p. 96] that the values of \bar{k} on the set

$$X = (Z/64Z) \cap (7 + 8Z) = \{7, 15, 23, 31, 39, 47, 55, 63\}$$

are determined by its values on the rest of $Z/64Z$.

Use the initial permutation [DE82, pp. 91-97; KO81, pp. 240-249; ME82, p. 155-160] IP and the bit-selection table [DE82, pp. 92-94; KO81, pp. 241-242; ME82, pp. 156-160] E to form a modified message

$$m : Z/96Z \times Z/2Z \rightarrow Z/2Z$$

i.e. a member of the set Δ of toroidal 96 by 2 matrices of zeros and ones.

The modified message m (which we will call a DES internal message) is formed from \bar{m} by means of a pure diffusion operation

$$\pi : D \rightarrow Z/64Z,$$

followed by multiplication by a constant matrix $w \in \Delta$, so that

$$m = w * (\bar{m} \circ \pi).$$

The transition from \bar{m} to m by means of the initial message diffusion π and the constant matrix w is key-independent and has no secrecy aspect. In other words \bar{m} may be secret but does not depend on \bar{k} . But π is neither secret nor dependent on \bar{m} or \bar{k} . The surjection π is

naturally associated with a certain 64 dimensional vector subspace Π of the 192 dimensional vector space Δ .

The map π is a surjection but not an injection. Therefore [MA67, p. 9] it has no left inverse function but has many right inverse functions. Using the IP^{-1} map [DE82, p. 92] we can easily fix upon a distinguished member of this set of right inverses, call it π^{-1} , which faithfully represents the map IP^{-1} , and which correctly reformats messages after the sixteen round operation of DES.

Independently of all this initial reformatting of the plaintext message \overline{m} so as to produce m , use the permuted choices (or so-called key permutations) [DE82, pp. 96-97; KO81, pp. 245-247; ME82, pp. 153-160] PC-1 and PC-2 in conjunction with the key schedule of left shifts [DE82, pp. 96-97; KO81, pp. 245-247; ME82, pp. 153-160] to turn the key \overline{k} into a modified key

$$k : Z/16Z \rightarrow \Delta ,$$

i.e. a list $(k[0], k[1], \dots, k[15])$ of sixteen 96 by 2 toroidal matrices. This modified key k (which we will call a DES internal key) is formed from (the external key) \overline{k} by means of sixteen pure diffusion operations

$$\phi[i] : Z/96Z \times Z/2Z \rightarrow Z/64Z,$$

$i \in Z/16Z$, and a constant matrix $v \in \Delta$ so that

$$k[i] = v * (\overline{k} \circ \phi[i])$$

We thus write k as a list

$$\begin{aligned} k &= (k[0], k[1], \dots, k[15]) \\ &= (v * (\bar{k} \circ \phi[0]), v * (\bar{k} \circ \phi[1]), \dots, v * (\bar{k} \circ \phi[15])) \end{aligned}$$

of sixteen members of Δ . The sixteen functions $\phi[0], \phi[1], \dots, \phi[15]$ are all naturally associated with a certain 48 dimensional vector subspace Φ of Δ .

The transition from \bar{k} to k by means of the initial key diffusion $\phi[i]$ and the constant matrix v has no secrecy aspect. In other words \bar{k} may be secret, but does not depend on \bar{m} . Moreover none of $v, \phi[0], \phi[1], \dots, \phi[15]$ are either secret or dependent on \bar{m} or \bar{k} .

At this point we have $m \in \Delta$ and $k \in \Delta^{Z/16Z}$. With these seventeen 96 by 2 toroidal matrices of zeros and ones at our disposal we can describe the 16-round internal structure of DES very simply. Note that everything done so far is possible without performing any rounds of the DES. It depends only on the message block \bar{m} and the key \bar{k} .

The round [DE82, pp. 92-96; KO81, pp. 240-248; ME82, pp. 141-142, 156-160] in DES is a map

$$\rho : \Phi \times \Pi \rightarrow \Pi$$

with the property that the restriction $\rho|_f$ of ρ to $\{f\} \times \Pi$ is (well, amounts to, in the obvious fashion) a permutation of Π for every

matrix $f \in \Phi$. We can say, if we choose, that the round ρ of DES is a family

$$\rho = \{\rho|_f : f \in \Phi\}$$

of replacements of Π .

The round ρ can be further analyzed. In fact,

$$\rho(x, y) = u * (y \circ \alpha) + (\sigma(x + v * y)) \circ \alpha$$

for every $x, y \in \Delta$. Here the plus sign $+$ denotes the natural vector space addition on the vector space Δ . Just add entrywise modulo 2. The times sign $*$ denotes entrywise multiplication (not matrix multiplication) of 96 by 2 toroidal matrices. The map

$$\sigma : \Delta \rightarrow \Delta$$

is a replacement corresponding to the action of the S-boxes [DE82, pp. 92-96; KO81, pp. 243-244]. The range Σ of σ is a 64 dimensional vector subspace of the 192 dimensional vector space Δ . The map

$$\alpha : D \rightarrow D$$

is a diffusions, i.e. is a self-map of the 192-element set D of ordered pairs which constitutes the domain of a modified message $m \in \Delta$. The matrix $u \in \Delta$ is a constant.

Note, at this point, that this description of DES does not speak of 16 rounds. There is just the round ρ . The round ρ is done sixteen

times in succession with (presumably) different input pairs. But it is just one map, not a list of 16 maps. It has no secrecy aspect. It does not depend on \overline{m} or \overline{k} . The action of DES in the key-setting \overline{k} on the message \overline{m} is thus

$$[\rho(k[15], \rho(k[14], \rho(\dots, \rho(k[2], \rho(k[1], \rho(k[0], v * (\overline{m} \circ \pi)))) \dots))] \circ \pi^{-1}$$

where $v \in \Delta$ is a constant and

$$\rho(x, y) = u * (y \circ \alpha) + (\sigma(x + v * y)) \circ \alpha.$$

Let us make this more explicit. Start with three fixed 96 by 2 toroidal matrices

$$u : D \rightarrow Z/2Z$$

$$v : D \rightarrow Z/2Z$$

$$w : D \rightarrow Z/2Z$$

These three fixed members of Δ can be viewed as nullary operations on Δ . There is one fixed replacement

$$\sigma : \Delta \rightarrow \Delta.$$

It can be viewed as a unary operation on Δ . There are two fixed binary operations on Δ , namely

$$+ : \Delta \times \Delta \rightarrow \Delta$$

$$* : \Delta \times \Delta \rightarrow \Delta$$

There are seventeen fixed initial diffusions

$$\begin{aligned}\pi &: D \rightarrow Z/64Z \\ \phi[0] &: D \rightarrow Z/64Z \\ \phi[1] &: D \rightarrow Z/64Z \\ &\vdots \\ \phi[15] &: D \rightarrow Z/64Z\end{aligned}$$

There is one fixed terminal diffusion

$$\pi^{-1} : Z/64Z \rightarrow D.$$

Note that the injection π^{-1} is one of the many right inverses of the surjection π . There are no left inverses of π . There is an internal diffusion

$$\alpha : D \rightarrow D$$

which takes place internal to the round. There are no confusions, i.e. no selfmaps of the alphabet $Z/2Z$ which are composed on the left of any symbols such as $k, \bar{k}, m, \bar{m}, u, v, w$ or σ . We shall see, later that the diffusion α makes use of selfmaps of $Z/2Z$. However the $Z/2Z$ this self-map acts on is not the alphabet, but rather the second cartesian factor in the cartesian product

$$Z/96Z \times Z/2Z = D.$$

which constitutes the domain, not the codomain of a message. Hence these latter selfmaps are diffusions, not confusions.

To employ the \bar{k} key-setting of DES on the plaintext message \bar{m} , one proceeds as follows to build a list of 17 members of Δ , followed by one member of $Z/2Z^{Z/64Z}$:

$$q[0] = w * (\bar{m} \circ \pi);$$

$$q[1] = u * (q[0] \circ \alpha) + (\sigma(\bar{k} \circ \phi[0] + v * q[0])) \circ \alpha;$$

$$q[2] = u * (q[1] \circ \alpha) + (\sigma(\bar{k} \circ \phi[1] + v * q[1])) \circ \alpha;$$

$$\vdots$$

$$q[14] = u * (q[13] \circ \alpha) + (\sigma(\bar{k} \circ \phi[13] + v * q[13])) \circ \alpha;$$

$$q[15] = u * (q[14] \circ \alpha) + (\sigma(\bar{k} \circ \phi[14] + v * q[14])) \circ \alpha;$$

$$q[16] = u * (q[15] \circ \alpha) + (\sigma(\bar{k} \circ \phi[15] + v * q[15])) \circ \alpha$$

$$\bar{y} = q[16] \circ \pi^{-1}.$$

4. The initial permutation IP and its inverse

Permutations will be written as products of disjoint cycles. For example

$$\beta = (1, 5, 2, 3)(4, 6)(7) = (7)(4, 6)(5, 2, 3, 1)$$

is the function β such that: $\beta(1) = 5$; $\beta(2) = 3$; $\beta(3) = 1$; $\beta(4) = 6$; $\beta(5) = 2$; $\beta(6) = 4$; $\beta(7) = 7$;

The initial permutation IP [DE82, p. 92] can be factored [DA84, p. 190] into disjoint cycles of lengths 1,2,3 and 6 in the following fashion.

$$IP = \prod U[j],$$

where the product is over $j \in \{0, 1, 2, 3, 4, 5, 6, 8, 10, 11, 13, 18, 21, 42\}$,
and

$$U[0] = (0, 57, 54, 12, 27, 39)$$

$$U[1] = (1, 49, 52, 28, 31, 7)$$

$$U[2] = (2, 41, 50, 44, 26, 47)$$

$$U[3] = (3, 33, 48, 60, 30, 15)$$

$$U[4] = (4, 25, 55)$$

$$U[5] = (5, 17, 53, 20, 29, 23)$$

$$U[6] = (6, 9, 51, 36, 24, 63)$$

$$U[8] = (8, 59, 38)$$

$$U[10] = (10, 43, 34, 40, 58, 46)$$

$$U[11] = (11, 35, 32, 56, 62, 14)$$

$$U[13] = (13, 19, 37, 16, 61, 22)$$

$$U[18] = (18, 45)$$

$$U[21] = (21)$$

$$U[42] = (42).$$

[DA84, pp. 189-191] contains a very complete discussion of IP from a variety of viewpoints and we will not consider it further, other than to note that (4), (5) and (7) in [DA84, p. 190] all express IP and IP^{-1} in various ways in terms of $Z/2Z$ arithmetic, the group $\text{SYM}(Z/6Z)$ of symmetries of a 6-member set, and $GF(64)$

arithmetic.

5. The initial diffusions which turn a 64-bit plaintext message block \overline{m} into a DES internal message m .

Let

$$\begin{aligned} A &= [Z/96Z] \cap [(1 + 3Z) \cup (0 + 12Z) \cup (11 + 12Z)] \\ &= \{0, 1, 4, 7, 10, 11, 12, 13, 16, 19, 22, 23, 24, 25, 28, \dots, \\ &\quad 67, 70, 71, 72, 73, 76, 79, 82, 83, 84, 85, 88, 91, 94, 95\} \end{aligned}$$

$$D = Z/96Z \times Z/2Z$$

$$Q = A \times Z/2Z$$

$$G = A \times \{0\}$$

$$F = A \times \{1\}$$

$$L = [(Z/96Z) \cap (1 + 3Z)] \times \{1\}$$

$$X = (Z/64Z) \cap (1 + 3Z) \times \{1\}.$$

Then, clearly,

$$\text{cardinality } (A) = 32 + 8 + 8 = 48$$

$$\text{cardinality } (D) = 96 * 2 = 192$$

$$\text{cardinality } (Q) = 48 * 2 = 96$$

$$\text{cardinality } (G) = 48 * 1 = 48$$

$$\text{cardinality } (F) = 48 * 1 = 48$$

$$\text{cardinality } (L) = 32 * 1 = 32.$$

We define $\nu : F \rightarrow L$ by setting

$$\nu(f) = f \quad \text{if } f \in F$$

$$\nu((12t, 1)) = (12t - 2, 1)$$

$$\nu((12t - 1, 1)) = (12t + 1, 1)$$

if $t \in Z/8Z$. See Table 5.1 below.

It is evident that ν is a 3 to 2 surjection. We define several vector spaces over the field $GF(2) = Z/2Z$. Let

$$\Delta = (Z/2Z)^D$$

$$\Pi = \{d \in \Delta : d(i, j) = 0 \quad \text{if } i \notin A\}$$

$$\Gamma = \{q \in \Pi : q(i, j) = 0 \quad \text{if } j \neq 0\}$$

$$\Phi = \{q \in \Pi : q(i, j) = 0 \quad \text{if } j \neq 1\}$$

Thus Π is the vector subspace of Δ consisting of all 96 by 2 toroidal matrices whose support is Q . Similarly Γ is the vector subspace of Π consisting of matrices supported on $A \times \{0\}$, and Φ consists of all matrices supported on $A \times \{1\}$. Also we need

$$\hat{\Pi} = \{q \in \Pi : q(12t - 2, j) = q(12t, j) \quad \text{and}$$

$$q(12t - 1, j) = q(12t + 1, j) \text{ for every } t \in Z, \text{ every } j \in Z\}$$

$$\hat{\Gamma} = \hat{\Pi} \cap \Gamma$$

$$\hat{\Phi} = \hat{\Pi} \cap \Phi.$$

Clearly we have $\Pi = \Gamma \oplus \Phi$ and $\hat{\Pi} = \hat{\Gamma} \oplus \hat{\Phi}$. Table 5.2 below describes dimensionalities and subspace relationships among these 7 vector spaces.

We also need the masks w, u and v which turn members of Δ into members of Π, Γ and Φ respectively. The vector $w \in \Pi$ has as many entries equal to 1 as a member of Π can have, i.e.

$$\begin{aligned} w(i, j) &= 1 && \text{if } (i, j) \in A \times Z/2Z \\ &= 0 && \text{otherwise} \end{aligned}$$

Similarly $u \in \Gamma$,

$$\begin{aligned} u(i, j) &= 1 && \text{if } (i, j) \in A \times \{0\} \\ &= 0 && \text{otherwise} \end{aligned}$$

and $v \in \Phi$,

$$\begin{aligned} v(i, j) &= 1 && \text{if } (i, j) \in A \times \{1\} \\ &= 0 && \text{otherwise} \end{aligned}$$

Evidently

$$u * v = 0$$

$$u * w = u$$

$$v * w = v$$

$$u + v = w$$

Also, for any $d \in \Delta$ we have

$$u * d = d * u \in \Gamma$$

$$v * d = d * v \in \Phi$$

$$w * d = d * w \in \Pi$$

We will set up a bijection between $\hat{\Pi}$ and the space of all 64-bit plaintext DES words. Then we will proceed in the spirit of [DA84]

and do all further DES operations in Π . The larger vector space Δ arises naturally from an attempt to make the data expansion effected by the bit selection [DE82, p. 93] table E and the workings of the DES round more simple.

The initial [KO81, pp. 240-242] permutation IP and the bit-selection [DE82, pp. 93-94] table E are two of the diffusions used to reformat a 64-bit plaintext message block for internal use by DES. In the treatment below it will be part of the conversion of a plaintext message block

$$\overline{m} \in (Z/2Z)^{Z/64Z}$$

into an internal DES message $m \in \Delta$. Tables 5.3 and 5.4 below give the values of γ , $\overline{\pi}$, π , $\overline{m} \circ \pi = \overline{m} \circ IP \circ \overline{\pi}$, w and $m = w * (\overline{m} \circ \pi)$. All of them are displayed as 96 by 2 toroidal matrices.

The diffusion

$$\gamma : D \rightarrow D$$

is the identity permutation of D , represented as a matrix. It is shown to give the reader a clear picture of where the (j, \hat{j}) th entry of each of the matrices shown is located. The diffusion

$$\pi = IP \circ \overline{\pi} : D \rightarrow Z/64Z$$

is a 3-to-1 surjection, represented as a matrix The map

$$\overline{m} \circ \pi = \overline{m} \circ IP \circ \overline{\pi} : D \rightarrow Z/2Z$$

is a member of Δ , and is represented as a matrix. The nullary operation (i.e. constant, or mask) $w \in \Delta$ is represented as a matrix.

The entrywise product

$$p = w * (\overline{m} \circ \pi) = w * (\overline{m} \circ IP \circ \overline{\pi})$$

is represented as a matrix. An entry of this matrix $w * (\overline{m} \circ \pi)$ must be zero if the corresponding entry of w is zero. Other entries of $w * (\overline{m} \circ \pi)$ can also be zero (for example the (0,0)th entry of $w * (\overline{m} \circ \pi)$ is zero if $\overline{m}(7) = 0$). Its left column consists of the entries indexed by indices of the form $(0, j) \in D$, and amounts to a 48-bit left-half word. Its right column consists of the entries indexed by pairs of the form $(1, j) \in D$, and amounts to a 48-bit right half word. There are relationships among its rows. Thus

$$\text{row } 0 = \text{row } 94$$

$$\text{row } 12 = \text{row } 10$$

$$\text{row } 24 = \text{row } 22$$

$$\vdots$$

$$\text{row } 72 = \text{row } 70$$

$$\text{row } 84 = \text{row } 82$$

also

$$\text{row } 11 = \text{row } 13$$

$$\text{row } 23 = \text{row } 25$$

row 35 = row 37

\vdots

row 83 = row 85

row 95 = row 1

Hence 32 of the rows of $w * (\overline{m} \circ \pi)$ determine all its rows. See [DA84, pp. 191-192] for an arithmetical description of the bit selection table E . Our approach is similar but we spread the bits of the initial 64-bit message more uniformly through a larger array.

We note that $\overline{\pi}$ and $\pi = IP \circ \overline{\pi}$ are single matrices. But the collection

$$\{w * (\overline{m} \circ \pi) : \overline{m} \in (Z/2Z)^{Z/64Z}\}$$

is a 64 dimensional subspace of Δ .

$$\nu(0, 1) = (94, 1)$$

$$\nu(1, 1) = (1, 1)$$

$$\nu(4, 1) = (4, 1)$$

$$\nu(7, 1) = (7, 1)$$

$$\nu(10, 1) = (10, 1)$$

$$\nu(11, 1) = (13, 1)$$

$$\nu(12, 1) = (10, 1)$$

$$\nu(13, 1) = (13, 1)$$

$$\nu(16, 1) = (16, 1)$$

$$\nu(19, 1) = (19, 1)$$

$$\nu(22, 1) = (22, 1)$$

$$\nu(23, 1) = (25, 1)$$

$$\nu(24, 1) = (22, 1)$$

$$\nu(25, 1) = (25, 1)$$

$$\nu(28, 1) = (28, 1)$$

$$\vdots$$

$$\nu(83, 1) = (85, 1)$$

$$\nu(84, 1) = (82, 1)$$

$$\nu(85, 1) = (85, 1)$$

$$\nu(88, 1) = (88, 1)$$

$$\nu(91, 1) = (91, 1)$$

$$\nu(94, 1) = (94, 1)$$

$$\nu(95, 1) = (1, 1)$$

Table 5.1. The 3 to 2 surjection $\nu : F \rightarrow L$

space	dimension							
	of space	at left	Is the space at left a subspace of the space below?					
Δ	192	yes						
Π	96	yes	yes					
$\hat{\Pi}$	64	yes	yes	yes				
Γ	48	yes	yes		yes			
$\hat{\Gamma}$	32	yes	yes	yes	yes	yes		
Φ	48	yes	yes				yes	
$\hat{\Phi}$	32	yes	yes	yes			yes	yes
		Δ	Π	$\hat{\Pi}$	Γ	$\hat{\Gamma}$	Φ	$\hat{\Phi}$

Table 5.2

$(0, 0)$	$(1, 0)$	31	63	7	6
$(0, 1)$	$(1, 1)$	0	32	57	56
$(0, 2)$	$(1, 2)$	31	63	7	6
$(0, 3)$	$(1, 3)$	0	32	57	56
$(0, 4)$	$(1, 4)$	1	33	49	48
$(0, 5)$	$(1, 5)$	2	34	41	40
$(0, 6)$	$(1, 6)$	1	33	49	48
$(0, 7)$	$(1, 7)$	2	34	41	40
$(0, 8)$	$(1, 8)$	1	33	49	48
$(0, 9)$	$(1, 9)$	2	34	41	40
$(0, 10)$	$(1, 10)$	3	35	33	32
$(0, 11)$	$(1, 11)$	4	36	25	24
$(0, 12)$	$(1, 12)$	3	35	33	32
$(0, 13)$	$(1, 13)$	4	36	25	24
$(0, 14)$	$(1, 14)$	3	35	33	32
$(0, 15)$	$(1, 15)$	4	36	25	24
$(0, 16)$	$(1, 16)$	5	37	17	16
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$(0, 81)$	$(1, 81)$	26	58	45	44
$(0, 82)$	$(1, 82)$	27	59	39	38
$(0, 83)$	$(1, 83)$	28	60	31	30
$(0, 84)$	$(1, 84)$	27	61	39	38
$(0, 85)$	$(1, 85)$	28	60	31	30
$(0, 86)$	$(1, 86)$	27	61	39	38
$(0, 87)$	$(1, 87)$	28	60	31	30
$(0, 88)$	$(1, 88)$	29	61	23	22
$(0, 89)$	$(1, 89)$	30	62	15	14
$(0, 90)$	$(1, 90)$	29	63	23	22
$(0, 91)$	$(1, 91)$	30	62	15	14
$(0, 92)$	$(1, 92)$	29	63	23	22
$(0, 93)$	$(1, 93)$	30	62	15	14
$(0, 94)$	$(1, 94)$	31	63	7	6
$(0, 95)$	$(1, 95)$	0	32	57	56

γ , the identity on D $\bar{\pi}$ $\pi = IP \circ \bar{\pi}$

Table 5.3

$\overline{m}(7)$	$\overline{m}(6)$	1	1	$\overline{m}(7)$	$\overline{m}(6)$
$\overline{m}(57)$	$\overline{m}(56)$	1	1	$\overline{m}(57)$	$\overline{m}(56)$
$\overline{m}(7)$	$\overline{m}(6)$	0	0	0	0
$\overline{m}(57)$	$\overline{m}(56)$	0	0	0	0
$\overline{m}(49)$	$\overline{m}(48)$	1	1	$\overline{m}(49)$	$\overline{m}(48)$
$\overline{m}(41)$	$\overline{m}(40)$	0	0	0	0
$\overline{m}(49)$	$\overline{m}(48)$	0	0	0	0
$\overline{m}(41)$	$\overline{m}(40)$	1	1	$\overline{m}(41)$	$\overline{m}(40)$
$\overline{m}(49)$	$\overline{m}(48)$	0	0	0	0
$\overline{m}(41)$	$\overline{m}(40)$	0	0	0	0
$\overline{m}(33)$	$\overline{m}(32)$	1	1	$\overline{m}(33)$	$\overline{m}(32)$
$\overline{m}(25)$	$\overline{m}(24)$	1	1	$\overline{m}(25)$	$\overline{m}(24)$
$\overline{m}(33)$	$\overline{m}(32)$	1	1	$\overline{m}(33)$	$\overline{m}(32)$
$\overline{m}(25)$	$\overline{m}(24)$	1	1	$\overline{m}(25)$	$\overline{m}(24)$
$\overline{m}(33)$	$\overline{m}(32)$	0	0	0	0
$\overline{m}(25)$	$\overline{m}(24)$	0	0	0	0
$\overline{m}(17)$	$\overline{m}(16)$	1	1	$\overline{m}(17)$	$\overline{m}(16)$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$\overline{m}(45)$	$\overline{m}(44)$	0	0	0	0
$\overline{m}(39)$	$\overline{m}(38)$	1	1	$\overline{m}(34)$	$\overline{m}(38)$
$\overline{m}(31)$	$\overline{m}(30)$	1	1	$\overline{m}(31)$	$\overline{m}(30)$
$\overline{m}(39)$	$\overline{m}(38)$	1	1	$\overline{m}(39)$	$\overline{m}(38)$
$\overline{m}(31)$	$\overline{m}(30)$	1	1	$\overline{m}(31)$	$\overline{m}(30)$
$\overline{m}(39)$	$\overline{m}(38)$	0	0	0	0
$\overline{m}(31)$	$\overline{m}(30)$	0	0	0	0
$\overline{m}(23)$	$\overline{m}(22)$	1	1	$\overline{m}(23)$	$\overline{m}(22)$
$\overline{m}(15)$	$\overline{m}(14)$	0	0	0	0
$\overline{m}(23)$	$\overline{m}(22)$	0	0	0	0
$\overline{m}(15)$	$\overline{m}(14)$	1	1	$\overline{m}(15)$	$\overline{m}(14)$
$\overline{m}(23)$	$\overline{m}(22)$	0	0	0	0
$\overline{m}(15)$	$\overline{m}(14)$	0	0	0	0
$\overline{m}(7)$	$\overline{m}(6)$	1	1	$\overline{m}(7)$	$\overline{m}(6)$
$\overline{m}(57)$	$\overline{m}(56)$	1	1	$\overline{m}(57)$	$\overline{m}(56)$

$$\overline{m} \circ \pi = \overline{m} \circ IP \circ \overline{\pi} \quad w \quad w * (\overline{m} \circ IP \circ \overline{\pi})$$

Table 5.4

6. The initial diffusions which turn a 64-bit external key block \bar{k} into a DES list of k sixteen internal keys.

The permuted [KO81, pp. 245-247] choices $PC - 1$ and $PC - 2$ are initial diffusions which will be used in this paper to help turn a 56 bit external DES key block

$$\bar{k} \subseteq (Z/2Z)^{Z/64Z}$$

into a list

$$k = (k[0], k[1], \dots, k[15])$$

of sixteen internal DES keys belonging to the 48 dimensional vector subspace Φ of the 192 dimensional vector space Δ . We will follow [DE82, p. 96] in regarding $PC - 1$ as an injection of the 56 member set $Z/64Z \setminus X$ into a 64 member set $Z/64Z$ rather than as a permutation of the 56-member set $Z/64Z \setminus X$. As always, however, we will follow [KO81] in starting our indexing with 0, rather than with 1. The table of DES key schedule shifts also plays a part in the process of converting a conventional DES key into a list of internal keys. It is necessary to perform several successive diffusions on a 64-bit DES key \bar{k} followed by an (entrywise) matrix multiplication, so as to produce an “internal key”, i.e. a list

$$k = (k[0], k[1], \dots, k[15])$$

of sixteen 96 by 2 toroidal matrices which will serve as key material in the internal format of the round structure of DES. For each $i \in$

$Z/16Z$ the internal i th key entry $k[i]$ will be a member of the 48 dimensional vector subspace Φ of the 192 dimensional vector space Δ of all 96 by 2 toroidal matrices over $GF(2) = Z/2Z$.

We start, therefore, with the DES internal key

$$\bar{k} = (\bar{k}(0), \bar{k}(1), \dots, \bar{k}(63))$$

and recall that it belongs to a 56 dimensional vector subspace of the 64 dimensional space of lists of 64 bits. This is because, as noted in Section 3, the bits $\bar{k}(7), \bar{k}(15), \dots, \bar{k}(63)$ are parity bits, whose values are determined by the other 56 bits of \bar{k} , the bits indexed by members of $Z/64Z \setminus X$.

The index set, $Z/28Z \times Z/2Z$, of the set of 28 by 2 toroidal matrices is important enough to have its own name. So we define

$$J = Z/28Z \times Z/2Z.$$

And we recall, from Section 3,

$$D = Z/96Z \times Z/2Z.$$

The first diffusion applied to \bar{k} is

$$\psi : J \rightarrow Z/64Z.$$

The diffusion ψ embodies the information contained in the permuted [DE82, p. 96] choice $PC - 1$. Once again [DA84, pp. 195-196]

describes $PC - 1$ in arithmetic terms and points out its simple structure, which a reader can easily discover in ψ . The diffusion ψ turns \bar{k} into a 28 by 2 toroidal matrix $\bar{k} \circ \psi$ over $Z/2Z$.

Then we have a list

$$\lambda = (\lambda[0], \lambda[1], \dots, \lambda[15])$$

of diffusions

$$\lambda[i] : J \rightarrow J$$

each of which replaces this 28 by 2 toroidal matrix $\bar{k} \circ \psi$ by a “left-shifted” version of itself (a phrase more faithful to the matrix picture would be “Ferris-wheeled”) induced by the key schedule [DE82, p.96] of left shifts LS. The index set for the list λ is, of course, $Z/16Z$.

Once the 16 member list

$$(\bar{k} \circ \psi \circ \lambda[0], \bar{k} \circ \psi \circ \lambda[1], \dots, \bar{k} \circ \psi \circ \lambda[15])$$

of 28 by 2 toroidal matrices over $Z/2Z$ has been constructed it is necessary to use a last key diffusion

$$\zeta : D \rightarrow J$$

to produce a list

$$(\bar{k} \circ \psi \circ \lambda[0] \circ \zeta, \bar{k} \circ \psi \circ \lambda[1] \circ \zeta, \dots, \bar{k} \circ \psi \circ \lambda[15] \circ \zeta)$$

of sixteen 96 by 2 toroidal matrices over $Z/2Z$. This diffusion ς embodies all the information contained in the key [DE82, p. 97] permutation $PC - 2$. Finally we must multiply (entrywise) each of these matrices by a “mask” matrix v which is zero in 144 of its entries, and has the value one only in those 48 entries corresponding to the 48 inputs to the S -boxes [DE82, pp. 92-97]. The matrix w is the nullary operation (mask) defined in Section 5. At this point we give the explicit characterizations of ψ , λ and ς . The toroidal matrices $\psi \in (Z/64Z)^J$ and $\varsigma \in J^D$ are shown in Figure 6.1. We have deliberately left three fourths of the entries of ς unevaluated (denoted by the sharp symbol $\#$). Any one of them can have any value in J the reader desires (such flexibility may lead to some simplification). This is because a mask v will be multiplied by the matrix we are building and will leave only zeros in these places in

$$k[i] = v * (\bar{k} \circ \psi \circ \lambda[i] \circ \varsigma) = v * (\bar{k} \circ \psi[i]) \in \Phi \subseteq \Delta$$

anyway.

For each $i \in Z/16Z$ the diffusion

$$\lambda[i] : J \rightarrow J$$

is defined by setting

$$\lambda[i](a, b) = (a + \ell(i), b),$$

where the 16-entry list ℓ of positive integers is given by

$$\begin{aligned}\ell &= (\ell(0), \ell(1), \dots, \ell(15)) \\ &= (1, 2, 4, 6, 8, 10, 12, 14, 15, 17, 19, 21, 23, 25, 27, 28).\end{aligned}$$

These successive positive integers are just the successive partial sums of the numbers of left shift positions in [DE82, p. 96]. Note that after 16 rounds the 28 by 2 toroidal matrix $k \circ \psi$ has been rolled all the way around to its original position, so that no reset is needed before encrypting the next DES message \overline{m} in the same key \overline{k} . Note the sum, $a + \ell(i)$, above. To show that it would be wrong to use the difference, $a - \ell(i)$, we will work out Example 6.1 below. Now it merely remains to multiply by the mask $v \in \Phi$ so as to zero out the whole left column (the entries with second index 0) as well as half of the right column of $\overline{k} \circ \psi \circ \lambda[i] \circ \zeta$. We thus have

$$\begin{aligned}k[i] &= v * (\overline{k} \circ \psi \circ \lambda[i] \circ \zeta) \\ &= v * (\overline{k} \circ \phi[i]).\end{aligned}$$

Example 6.1: To verify that these diffusions actually faithfully represent the key schedule of DES let us follow k_8 , k_{44} and k_{29} in Konheim's [KO81, p. 247] notation. Because we have kept the parity bits in positions 7 modulo 8 we have the correspondence

$$\begin{aligned}k_8 &= \overline{k}(9), \\ k_{29} &= \overline{k}(33), \\ k_{44} &= \overline{k}(50).\end{aligned}$$

We verify that

$$\psi((14, 0)) = 9$$

$$\psi((17, 0)) = 50$$

$$\psi((11, 0)) = 33$$

and that

$$\lambda[1]((13, 0)) = (14, 0)$$

$$\lambda[1]((16, 0)) = (17, 0)$$

$$\lambda[1]((10, 0)) = (11, 0)$$

and that

$$\varsigma((0, 0)) = (13, 0)$$

$$\varsigma((1, 0)) = (16, 0)$$

$$\varsigma((4, 0)) = (10, 0)$$

Hence

$$\begin{aligned} \bar{k} \circ \psi \circ \lambda[1] \circ \varsigma)((0, 0)) &= \bar{k}(\psi(\lambda[1](\varsigma((0, 0)))) \\ &= \bar{k}(\psi(\lambda[1]((13, 0)))) \\ &= \bar{k}(\psi((14, 0))) \\ &= \bar{k}(9) \\ &= k_8 \end{aligned}$$

and

$$\begin{aligned} (\bar{k} \circ \psi \circ \lambda[1] \circ \varsigma)((0, 0)) &= \bar{k}(\psi(\lambda[1](\varsigma((1, 0)))) \\ &= \bar{k}(\psi(\lambda[1]((16, 0)))) \\ &= \bar{k}(\psi((17, 0))) \\ &= \bar{k}(50) \\ &= k_{44} \end{aligned}$$

and

$$\begin{aligned}
 (\bar{k} \circ \psi \circ \lambda[1] \circ \varsigma)((4, 0)) &= \bar{k}(\psi(\lambda[1](\varsigma((4, 0)))) \\
 &= \bar{k}(\psi(\lambda[1]((10, 0)))) \\
 &= \bar{k}(\psi((11, 0))) \\
 &= \bar{k}(33) \\
 &= k_{29}
 \end{aligned}$$

And this, of course, is what can be found in [KO81, p. 247] as the beginning of the key used in the first round of DES.

56	62
48	54
40	46
32	38
24	30
16	22
8	14
0	6
57	61
49	53
41	45
33	37
25	29
17	21
9	13
1	5
58	60
50	52
42	44
34	36
26	28
18	20
10	12
2	4
59	27
51	19
43	11
35	3

 ψ

Figure 6.1

# (13,1)	⋮	⋮
# (16,1)	# #	# (22,1)
# #	# #	# #
# #	# (25,1)	# #
# (10,1)	# (7,1)	# #
# #	# (15,1)	# (16,1)
# #	# (6,1)	# #
# (23,1)	# #	# #
# #	# #	# (4,1)
# #	# (26,1)	# (19,1)
# (0,1)	# #	# (15,1)
# (4,1)	# #	# (20,1)
# (2,1)	# (19,1)	# #
# (27,1)	# #	# #
# #	# #	# (10,1)
# #	# (12,1)	# #
# (14,1)	# (1,1)	# #
# #	# (12,1)	# (27,1)
# #	# (23,1)	# #
# (5,1)	# #	# #
# #	# #	# (5,1)
# #	# (2,1)	# (24,1)
# (20,1)	# #	# (17,1)
# (9,1)	# #	# (13,1)
# (22,1)	# (8,1)	# #
# (18,1)	# #	# #
# #	# #	# (21,1)
# #	# (18,1)	# #
# (11,1)	# (26,1)	# #
# #	# (1,1)	# (7,1)
# #	# (11,1)	# #
# (3,1)	# #	# #
⋮	# #	# (0,1)
	⋮	# (3,1)

Figure 6.2

The top,middle, and bottom thirds
of the 96 by 2 toroidal matrix ζ

7. The DES round ρ , in which an internal key k interacts with an internal message m .

The DES wire-crossing [DE82, p. 93; KO81, p. 245] P and the selection [DE82, p. 94] functions, i.e. S -boxes [KO81, p. 244] are used in each of the sixteen actions of the DES round. We now see that an internal message m and an entry $k[i]$ of an internal key list k are members of Δ . In fact

$$m \in \hat{\Pi} \subseteq \Pi \subseteq \Delta$$

$$k[i] \in \Phi \subseteq \Pi \subseteq \Delta.$$

The round ρ of DES proceeds as follows. The mask v is such that

$$v * m \in \hat{\Phi} \subseteq \Phi.$$

Hence

$$v * m + k[i] \in \Phi.$$

This vector $v * m + k[i]$ is input to the replacement σ corresponding to the S -boxes [KO91, p. 244] and, after wire crossing [KO81, p. 245] and masking, comes out as a member δ of $\hat{\Gamma}$. Meanwhile $u * m$ (a member of $\hat{\Gamma}$) is diffused by a column interchange to produce a member $\bar{\delta}$ of $\hat{\Phi}$. The matrix

$$\delta \oplus \bar{\delta} \in \hat{\Pi} = \hat{\Gamma} \oplus \hat{\Phi}$$

is the result of the round ρ .

We now carry out this process in detail.

In detail the process is as follows. Before the first action of the round ρ there is an initial internal message $m \in \Pi$. Clearly, then the (entrywise) product satisfies

$$v * m \in \Phi.$$

Also there is an entry $k[0]$ of the internal key k . It satisfies

$$k[0] \in \Phi.$$

Consequently their (entrywise) sum also belongs to the 48 dimensional vector space Φ , i.e.

$$v * m + k[0] \in \Phi.$$

We have a choice as to how we view the action of the S -boxes in the context of Δ . We can regard this action as a replacement of Δ (i.e. as a function with domain and codomain both equal to Δ) which is independent of 144 of the 192 entries of a matrix

$$v * m + k[0] = y \in \Delta.$$

We can also regard it as a function from Φ to Φ , to be followed by a diffusion corresponding to wire crossing and interchange of right half and left half words. This latter approach seems more in keeping with the standard descriptions of DES and we will adopt it.

So we will start by writing

$$\Phi = \Phi[0] \oplus \Phi[1] \oplus \dots \oplus \Phi[7]$$

$$\hat{\Phi} = \hat{\Phi}[0] \oplus \hat{\Phi}[1] \oplus \dots \oplus \hat{\Phi}[7]$$

where each $\Phi[i]$ is 6 dimensional, each $\hat{\Phi}[i]$ is a 4 dimensional subspace of $\Phi[i]$ and, in fact

$$\Phi[0] = \{t \in \Delta : t(i, j) = 0 \text{ unless } j = 1$$

$$\text{and } i \in \{0, 1, 4, 7, 10, 11\}\}$$

$$\hat{\Phi}[0] = \{t \in \Phi[0] : t(0, 1) = t(11, 1) = 0\}$$

$$\Phi[1] = \{t \in \Delta : t(i, j) = 0 \text{ unless } j = 1$$

$$\text{and } i \in \{12, 13, 16, 19, 22, 23\}\}$$

$$\hat{\Phi}[1] = \{t \in \Phi[1] : t(12, 1) = t(23, 1) = 0\}$$

$$\vdots$$

$$\Phi[7] = \{t \in \Delta : t(i, j) = 0 \text{ unless } j = 1$$

$$\text{and } i \in \{84, 85, 88, 91, 94, 95\}\}$$

$$\hat{\Phi}[7] = \{t \in \Delta : t(84, 1) = t(95, 1) = 0\}.$$

The first (i.e. zeroth) S -box determines a map

$$\sigma[0] : \Phi[0] \rightarrow \hat{\Phi}[0]$$

and similarly

$$\sigma[i] : \Phi[i] \rightarrow \hat{\Phi}[i]$$

for $0 \leq i \leq 7$. We will not describe these individual S -box maps any further. The nonlinear heart of DES is thus based on the map

$$\sigma[0] \oplus \sigma[1] \oplus \dots \oplus \sigma[7] = \bar{\sigma} : \Phi \rightarrow \hat{\Phi} \subseteq \Phi$$

Evidently the unary operation $\bar{\sigma}$ is a replacement of Φ . Its working is

$$\bar{\sigma}(f) = (\sigma[0] \oplus \dots \oplus \sigma[7])(f[0] \oplus \dots \oplus f[7]) = \sigma[0](f[0]) \oplus \dots \oplus \sigma[7](f[7]).$$

In other words each S -box works separately on its 6-bit input to produce its 4-bit output.

The support of $f \in \Delta$ is the 48 member set F , whereas the support of $\bar{\sigma}(f) \in \Delta$ is the 32 member subset L of F . To turn the wire crossing [DE82, p. 93; KO81, p. 245] P to a diffusion which permutes L we introduce the permutation

$$\bar{\mu} = \bar{\bar{\mu}}[0]\bar{\bar{\mu}}[1]$$

of $Z/32Z$ where

$$\bar{\bar{\mu}}[0] = (0, 15, 9, 14, 30, 3, 20, 31, 24, 18, 23, 8)$$

$$\bar{\bar{\mu}}[1] = (1, 6, 27, 5, 11, 25, 12, 4, 28, 21, 26, 29, 10, 22, 2, 19, 13, 17, 7, 16)$$

It is easy to see [DE82, p. 93; KO81, p. 245] that $\bar{\mu}$ embodies the post S -box wire crossing P and that we use it to produce the diffusion

$$\mu : D \rightarrow D$$

such that

$$\mu(1 + 3i, 1) = (1 + 3\overline{\mu}(i), 1)$$

if $(1 + 3i, 1) \in L$ and

$$\mu(j, k) = (j, k)$$

if $(j, k) \notin L$. After this we need the standard diffusion which splits L so as to cover F , i.e. the map

$$\nu : F \rightarrow L$$

defined in Section 5 above.

We also need the “column interchange” (i.e. interchange of left and right half-words) diffusion

$$\alpha : D \rightarrow D$$

defined by setting

$$\alpha((i, j)) = (i, j + 1)$$

since $D = Z/96Z \times Z/2Z$ the addition takes place in $Z/2Z$ and amounts to the permutation $(0, 1)$ of the set $\{0, 1\}$.

The round of DES thus takes $m \in \Delta$, and splits it into $u * m \in \Gamma$ and $v * m \in \Phi$ in the sense that

$$u \in \Gamma$$

$$v \in \Phi$$

$$u * m + v * m = (u + v) * m = m \in \Delta .$$

Then $k[i]$ is added to $v * m$ to yield

$$k[i] + v * m \in \Phi .$$

The replacement $\bar{\sigma} : \Phi \rightarrow \Phi$ is then applied to yield

$$\bar{\sigma}(v * m + k[i]) \in \Phi$$

$$\bar{\sigma}(k[i] + v * m) \in \Phi .$$

Then the two diffusions

$$\mu : D \rightarrow D$$

$$\nu : f \rightarrow L$$

are applied to $\bar{\sigma}(k[i] + v * m)$ to yield

$$\sigma(k[i] + v * m) = (\bar{\sigma}(k[i] + v * m)) \circ \mu \circ \nu \in \Gamma$$

and α is applied to m and to $\sigma(k[i] + v * m)$ to yield

$$m \circ \alpha \in \Delta$$

$$\sigma(k[i] + v * m) \circ \alpha \in \Gamma$$

Then $m \circ \alpha$ is masked by $u \in \Phi$ to yield

$$u * (m \circ \alpha)$$

Finally, an addition produces

$$\begin{aligned}
 & u * (m \circ \alpha) + (\bar{\sigma}(k[i] + v * m)) \circ \mu \circ \nu \circ \alpha \\
 &= u * (m \circ \alpha) + \sigma(k[i] + v * m) \circ \alpha \\
 &= \rho(k[i], m).
 \end{aligned}$$

8. The terminal diffusion π^{-1} which produces a cryptext message in 64-bit block form.

The final [DE82, p. 92] permutation IP^{-1} is one of the diffusions used to reformat an internal DES message after the sixteenth operation of the round so as to produce a correctly formatted 64-bit cryptext message block. Consider the injection

$$\pi^{-1} : Z/64Z \rightarrow D$$

defined by setting

$$\pi^{-1} = (IP^{-1}(3t + 1), 0)$$

if $0 \leq t \leq 31$, and

$$\pi^{-1} = (IP^{-1}(32 + 3t + 1), 1)$$

if $32 \leq t \leq 63$. It is easy to verify that $\pi \circ \pi^{-1}$ is the identity function on $Z/64Z$.

9. Recap of DES from the confusion/diffusion/arithmetic viewpoint.

It is clear from the foregoing that DES used only diffusion and replacement, no confusion. We thus seem, on a superficial reading, to be at odds with [DA84, p. 187] when those authors speak of “a representation of the DES as a cascade of substitutions and permutations.” But this surface appearance of conflict is only because they are using intuitively plausible terminology, whereas we have set confusion (hence substitution) in a rigorous context which banishes replacement (hence the action of the S -boxes) to the realm of arithmetic. This is, in turn, true because we have explicitly defined the alphabet of symbols which DES uses, namely the 2-letter alphabet

$$\{0, 1\} = GF(2) = \mathbb{Z}/2\mathbb{Z},$$

and have, consequently been forced to choose

$$P = \mathbb{Z}/64\mathbb{Z}$$

as the set of letter positions in a 64-bit “message”. The reader can object that the alphabet could be taken as the set of all $A = (\mathbb{Z}/2\mathbb{Z})^{(\mathbb{Z}/64\mathbb{Z})}$ 64-bit words. But at that level DES would merely be a simple substitution cipher, and no deeper analysis would be called for. What about regarding DES words as lists of sixteen

4-bit words, i.e. choosing

$$P = Z/16Z$$

$$A = (Z/2Z)^{Z/4Z} ?$$

Neither we nor [DA84] have devoted any space to explicit consideration of such a formulation of the DES, though it might prove interesting.

Why didn't its designers put any confusion into DES? For one thing, the alphabet A used by DES is the field

$$A = GF(2) = Z/2Z.$$

Since A has only 2 members, we see that $\text{SYM}(A)$ has only 2 members, A^A has only 4 members, and even $2^{A \times A}$ has only 16 members. A cryptosystem designer with only 16 confusion maps at his disposal doesn't have much running room and might be inclined to abandon the confusion approach for that reason. He could, however, fall back on a large family (i.e. a family determined by a large index set I)

$$f : I \rightarrow 2^{A \times A}$$

of binary relations on $A = Z/2Z$. One attractive possibility is a polyalphabetic substitution cipher in the sense made precise in [BL85, pp. 322-326].

Another reason for shunning confusion in DES could be that diffusion is cryptographically stronger, in a sense, on messages belonging to $(Z/2Z)^G$, where G is a group of reasonably large order.

Consider a known plaintext attack on a 16-alphabetic substitution cipher acting on 16 bit messages

$$m \in (Z/2Z)^{(Z/16Z)}.$$

If the ciphertext version of

$$m = (1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)$$

is m itself then all 16 alphabets have been recovered and the cryptanalyst has completely broken the cipher (i.e. has narrowed the original 2^{16} possible polyalphabetic cipher keys down to 1). But if she is dealing with a transposition cipher and finds that the above message m is encrypted as itself under the cipher, she has merely narrowed an original $16!$ possible cipher keys down to $(8!)^2 = 16!/12,870$ possible keys. So she has both a smaller reduction factor (12,870 vs. 65,536) and a larger remaining collection of possible keys.

The expansion of perspective in this paper from lists of 64 bits to members of the vector space Δ of 96 by 2 toroidal matrices over $Z/2Z = GF(2)$ simplified the description of the operation of the bit selection table E [DE82, p. 93; KO81, p. 242]. Further expansion of the size of the vector space beyond 192 dimensions can be used to simplify the description of key diffusions and, perhaps, S -boxes. The question is where the optimum stopping place lies. This would be a vector space within which most operations are very simple, but yet a space not too large to admit of manipulation by a cryptanalyst.

There are precedents for such an expansion of viewpoint in the success of tensor product methods in algebra and geometry. One example would be the use of multilinear maps on $R^n \times R^n \times \dots \times R^n$ to define polynomial maps on R^n . It remains to be seen to what extent a comparable approach will benefit cryptosystem design or cryptanalysis.

By this time the general features of the confusion/diffusion arithmetic approach to cryptography begun in [BL85b] are fairly clear. In DES we see quite a lot of simple arithmetic of binary operations (e.g., group addition modulo 2 or modulo 28, monoid multiplication modulo 2) and of nullary operations (such as the constant matrices u, v and w belonging to the vector space Δ) as well as a little fancy (and expensive) arithmetic of unary operations (the map σ corresponding to the S -boxes, some expansions and wire crossing) and a lot of diffusion. Most of our diffusions were, in fact, functions. Indeed most were either injections or surjections.

We hope at this point, to have clarified for the reader all the wire crossings, tables, boxes, (so called) substitutions which are really replacements, permutations which aren't really permutations, left shifts, schedules, half words (which are merely columns of matrices), blocks.

Employment of the methodology of this paper makes it possible to exorcise lugs, pins, rotors, shift registers, grilles, squares,

wheels, ... from other well-known cryptosystems. Not that these notions have served ill up to now – after all, many of them have been, or even still are, physically present and functioning in our crypto boxes, or grilles, or spools, or It's just that they are too many, too baroque, too far from the silicon medium and too unlike the mathematical notions which both builders and breakers employ in their work on cryptosystems. Also, of course, they have an unnecessarily finitist influence on our ways of speaking (hence thinking) about cryptography.

NSA Grant MCS 904-83-H-0002 supported this research.

10. References.

- BE82 H. Beker and F. Piper, *Cipher Systems: The Protection of Communications*, Wiley-Interscience, New York (1982).
- BL83 G. R. Blakley and Laif Swanson, Infinite structures in information theory, *Advances in Cryptology: Proceedings of Crypto '82*, Plenum Press (1983), pp. 39-50.
- BL85a G. R. Blakley and Catherine Meadows, Security of ramp schemes, in G. R. Blakley and D. Chaum, (editors), *Advances in Cryptology, Proceedings of Crypto '84*, Springer-Verlag, Berlin (1985), pp. 242-268.
- BL85b G. R. Blakley, Information theory without the finiteness assumption, I: Cryptosystems as group-theoretic objects, in

- G. R. Blakley and D. Chaum, (editors), *Advances in Cryptology, Proceedings of Crypto '84*, Springer-Verlag, Berlin (1985), pp. 314-338.
- BL87 G. R. Blakley and W. Rundell, A cryptosystem based on an analog of heat flow, Technical Report, September (1985).
- DA84 M. Davio, Y. Desmedt, M. Fosseprez, R. Govaerts, J. Hulsbosch, P. Neutjens, P. Piret, J. -J. Quisquater, J. Vandewalle and P. Wouters, Analytical Characteristics of the DES, in *Advances in Cryptology, Proceedings of Crypto '83*, D. Chaum, Editor, Plenum Press, New York (1984), pp. 171-202.
- DE82 D. E. R. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, Massachusetts (1980).
- DI79 W. Diffie and M. E. Hellman, Privacy and authentication, An introduction to cryptography, *Proceedings of the IEEE*, vol. 67 (1979), pp. 397-427.
- GR68 G. Grätzer, *Universal Algebra*, Van Nostrand, Princeton, New Jersey (1968).
- HA60 P. R. Halmos, *Naive Set Theory*, Van Nostrand, Princeton, New Jersey (1960).
- HO71 K. Hoffman and R. Kunze, *Linear Algebra, Second Edition*, Prentice Hall, Englewood Cliffs, New Jersey (1971).

- KI71 J. Killingbeck and G. H. A. Cole, *Mathematical Techniques and Physical Applications*, Academic Press, New York (1971).
- KO56 A. N. Kolmogoroff, On the Shannon theory of information transmission in the case of continuous signals, *IEEE Transactions on Information Theory*, vol. IT2 (1956), pp. 102-108.
- KO81 A. G. Konheim, *Cryptography: A Primer*, Wiley-Interscience, New York (1981).
- ME82 C. H. Meyer and S. M. Matyas, *Cryptography: A New Dimension in Computer Data Security*, Wiley-Interscience, New York (1982), Third Printing.
- LI83 R. Lidl and H. Niederreiter, *Finite Fields*, Volume 20 of the *Encyclopedia of Mathematics and its Applications*, Addison-Wesley, Reading, Massachusetts (1983).
- MA67 S. MacLane and G. Birkhoff, *Algebra*, Macmillan, New York (1967).
- MA78 F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam (1978).
- ME82 C. H. Meyer and S. M. Matyas, *Cryptography: A New*

Dimension in Computer Data Security, Wiley-Interscience, New York (1982).

MO63 G. D. Mostow, J. H. Sampson and J. -P. Meyer, Fundamental Structures of Algebra, McGraw-Hill, New York (1963).

NI59 H. K. Nickerson, D. C. Spencer and N. E. Steenrod, Advanced Calculus, Van Nostrand, Princeton, New Jersey (1959).

PA66 H. Paley and P. Weichsel, A First Course in Abstract Algebra, Holt, Rinehart and Winston, New York (1966).

RO64 G. -C. Rota, On the foundations of combinatorial theory, I. The theory of Möbius functions, Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete, Vol. 2 (1964), pp. 340-368.

Analysis of a Public Key Approach Based on Polynomial Substitution

Harriet Fell
Northeastern University
Boston, Massachusetts

Whitfield Diffie
Bell-Northern Research
Mountain View, California

1 Introduction

Ever since the discovery of public key cryptography in 1975², the search for public key cryptosystems has been a central theme of cryptographic research. The public key cryptosystems^{5,6,7} that have been investigated during this period, however, are slower than conventional systems, require more storage, and, being based on areas of mathematics that were not previously important in cryptography, have not inspired the same degree of trust as conventional systems. It would therefore be desirable to develop new techniques based on principles both different from those employed in current public key cryptosystems and more closely allied with conventional cryptography.

Several years ago, after the development of the public key concept, but before any plausible examples were known, a suggestion along these lines was made to one of the authors by John McCarthy of Stanford University, who said he had gotten the idea from talking with an algebraic geometer about birational transformations. The idea was to build inverse pairs of multivariate polynomial transformations by a procedure commonly employed in algebraic geometry to construct inverse pairs of rational transformations.

2 The Fundamental Scheme

Our approach is to regard the plaintext as an n -vector of elements selected from a suitable ring R and build an invertible polynomial transformation P of several variables from R^n to R^n . The coefficients of this transformation will be the public key and the inverse transformation Q the secret key. Thus:

$$\begin{array}{ccc} \text{Plaintext} & \xrightarrow{P} & \text{Ciphertext} \\ \bar{x} = (x_1, \dots, x_n) & \mapsto & (P_1(\bar{x}), \dots, P_n(\bar{x})) \end{array}$$

where $x_1, \dots, x_n \in R$ and P_1, \dots, P_n are multivariate polynomials with coefficients in R .

Assume, for example, that the plaintext is a vector of three components, x , y and z , from a ring R and that p_0 and p_1 are polynomials each in one variable over R . We can now build up a polynomial transformation of three variables by acting on the variables one at a time.

In the first round, (x, y, z) is carried to:

$$(x_1, y_1, z_1) = (x, y, z + p(x, y)).$$

where p can be either p_0 or p_1 . In the second, (x_1, y_1, z_1) goes to:

$$(x_2, y_2, z_2) = (x_1 + p(y_1, z_1), y_1, z_1).$$

The process continues:

$$(x_3, y_3, z_3) = (x_2, y_2 + p(x_2, z_2), z_2)$$

until after a number of rounds:

$$P(x, y, z) = (x_{k-1} + p(y_{k-1}, z_{k-1}), y_{k-1}, z_{k-1})$$

is a nonlinear, invertible, polynomial transformation on a module, M , of dimension 3 over R . The secret key is the sequence of choices of p_0 or p_1 and the order in which they are applied to x , y , and z . For example:

$$\begin{aligned}(x_1, y_1, z_1) &= (x + p_1(y, z), y, z) \\(x_2, y_2, z_2) &= (x_1, y_1, z_1 + p_0(x_1, y_1)) \\(x_3, y_3, z_3) &= (x_2, y_2 + p_0(x_2, z_2), z_2) \\(x_4, y_4, z_4) &= (x_3, y_3, z_3 + p_1(x_3, y_3)) \\P(x, y, z) &= (x_5, y_5, z_5) = (x_4 + p_0(y_4, z_4), y_4, z_4)\end{aligned}$$

The secret key is $((x, 1), (z, 0), (y, 0), (z, 1), (x, 0))$. The inverse transformation can be found by using the key in reverse: $((x, 0), (z, 1), (y, 0), (z, 0), (x, 1))$, i.e.:

$$\begin{aligned}(x_4, y_4, z_4) &= (x_5 - p_0(y_5, z_5), y_5, z_5) \\(x_3, y_3, z_3) &= (x_4, y_4, z_4 - p_1(x_4, y_4))\end{aligned}$$

and so on.

Naturally the number of polynomials need not be limited to 2 nor the dimension, d , of M over R to 3, but there is probably no virtue in using polynomials of degree other than $d - 1$.

This plan offers on its face, not only plausible hope for constructing inverse pairs of transformations, but one with very close ties to the shift register mathematics of conventional cryptography. The alternate transformation of variables x and y is closely analogous to alternate operation on the left and right halves in DES³. In general, the notion of modifying some components of a vector by adding to them functions of other components underlies all shift registers both linear and nonlinear^{1,4}.

A key difference between the construction of conventional cryptosystems and public key cryptosystems lies in the way the systems are presented to the user. A shift register cryptosystem is a description of the way in which the plaintext is modified incrementally through a number of iterations to become the ciphertext. Such a description precludes public key use because it can equally readily be read in the other direction as a description of how to derive the plaintext from the ciphertext by incremental modifications. In order to develop a public key system along these lines, it is necessary to simplify the equations that arise from the incremental substitution process in such a way as to conceal the substitutions.

On first glance, it seems sufficient to carry out the substitutions as the process goes on. On second, it becomes obvious that the number of coefficients in the polynomials will grow to astronomical proportions after only a few iterations. In order to prevent the equations from exploding into unusable bulk, some device must be found for eliminating most of the terms; the most obvious such devices are nilpotence and J-rings and these will be examined in the remainder of this paper.

3 Reducing the Number of Coefficients

3.1 Nilpotence

A ring R is nilpotent if there is an integer $k \geq 1$ such that $\{R\}^k = \{0\}$. That means that for any elements $r_1, \dots, r_k \in R$ the product $r_1 r_2 \cdots r_k$ is zero, so a multivariate polynomial with coefficients in R can only have meaningful terms whose total degree does not exceed k .

There is a hitch, however, in applying nilpotence in our fundamental scheme. We expect the scheme, described in section 2, to yield a transformation of the form:

$$\begin{array}{ccc} \text{plaintext} & \mathbf{P} & \text{ciphertext} \\ \bar{x} = (x_1, \dots, x_n) & \mapsto & (P_1(\bar{x}), \dots, P_n(\bar{x})) \end{array} \quad (3.1)$$

where $x_1, \dots, x_n \in R$ and P_1, \dots, P_n have coefficients in R . Note that no transformation of this form can be invertible. Suppose that each P_i has constant term C_i . Let $T_i(\bar{x}) = P_i(\bar{x}) - C_i$. The system \mathbf{P} is invertible if and only if the system $T = (T_1, \dots, T_n)$ is invertible but T can never be invertible as $T_i(\bar{x}) \in (R)^2 \subsetneq R$.

What went wrong is that although the iteration in section 2 always produces an invertible transformation, if we apply this iteration scheme in a ring without a unit, we do not get a transformation of the form (3.1) but rather one of the form:

$$\begin{array}{ccc} \text{plaintext} & \mathbf{P} & \text{ciphertext} \\ \bar{x} = (x_1, \dots, x_n) & \mapsto & \bar{x} + (P_1(\bar{x}), \dots, P_n(\bar{x})) \end{array} \quad (3.2)$$

Where $x_1, \dots, x_n \in R$ and P_1, \dots, P_n have coefficients in R . The transformation (3.1) is a polynomial transformation but its coefficients are not all in R as R is nilpotent and cannot contain a unit element.

We can still make use of nilpotence, however. We take R to be a finite local ring, that is a finite commutative ring with 1 that has a unique maximal ideal N of nilpotents. (Note R/N is a finite field.) The general form of the encryption transformation again becomes:

$$\begin{array}{ccc} \text{plaintext} & \mathbf{P} & \text{ciphertext} \\ \bar{x} = (x_1, \dots, x_n) & \mapsto & (P_1(\bar{x}), \dots, P_n(\bar{x})) \end{array} \quad (3.3)$$

where x_1, \dots, x_n and $P_1(\bar{x}), \dots, P_n(\bar{x})$ all lie in N . That is, \mathbf{P} is a multivariate polynomial transformation from R^n to R^n that is invariant and invertible on N^n . The number of coefficients in the polynomials is restricted by the nilpotence because terms of high total degree are identically zero on N^n and we do not care how they behave on the rest of R^n . The transformations generated by the iteration scheme of section 2 will be of this type if we choose the polynomials p_1, p_2, \dots to have coefficients in N or to have coefficients in R but constant terms in N .

3.2 J-Rings

Let R be a finite commutative ring. R is a J-ring if there is an integer $d \geq 1$ such that $a^d = a$ for any $a \in R$. A multivariate polynomial over a J-ring can be reduced to a polynomial all of whose terms have individual degrees $\leq d$ (or total degrees $< dn$ where n is the number of variables). This time the transformation will be of the form of equation (3.1).

3.3 Upper-Triangular Matrices

We represent the plaintext by a pair of upper-triangular matrices with entries from a finite ring R . The encryption transformation will be of the form:

$$\begin{array}{ccc} \text{plaintext} & \mathbf{P} & \text{ciphertext} \\ (X_1, X_2) & \mapsto & (P_1(X_1, X_2), P_2(X_1, X_2)) \end{array} \quad (3.4)$$

where X_1, X_2 are upper-triangular $k \times k$ matrices over R and P_1, P_2 have coefficients in R . If M_1, \dots, M_k are upper triangular $k \times k$ matrices over R then the product $M_1 \cdots M_k$ is zero. This means that the polynomials will have terms of total degree at most $k - 1$. As the matrices do not commute, there are more terms to deal with than in the commutative nilpotent case (3.1) but there is also hope that non-commutativity will make lower degree polynomial systems more difficult to invert.

	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=11	k=12	k=13	k=14	k=15	k=16	k=17	k=18
n=2	5 192 384	10 320 640	15 480 960	21 872 1344	28 898 1792	36 1152 2304	45 1440 2880	55 1760 3520	66 2112 4224	78 2496 4992	91 2912 5824	105 3360 6720	120 3840 7680	136 4352 8704	153 4896 9792	171 5472 10944	190 6080 12160
n=3	10 220 660	20 440 1320	35 770 2310	58 1232 3696	84 1848 5544	120 2640 7920	165 3630 10890										
n=4	15 240 960	35 560 2240	70 1120 4480	125 2016 8064	210 3360 13440												
n=5	21 273 1365	56 728 3640	125 1638 8190	252 3276 16380													
n=6	28 308 1848	84 924 5544	210 2310 13860														
n=7	36 360 2520	120 1200 8400	330 3300 23100														
n=8	45 360 2880	165 1320 10560															
n=9	55 440 3360	220 1760 15840															
n=10	66 462 4620	286 2002 20020															
n=11	78 462 5082	364 2184 24024															
n=12	91 546 6008	455 2730 32760															
n=13	105 525 6825	560 2800 36400															
n=14	120 600 8400	680 3400 47600															

For multivariate polynomials of degree k in n variables

126	$T(n, k) = \#$ of terms possible in one polynomial
2016	$\lceil \frac{64}{n} \rceil \cdot T(n, k) = \#$ of bits to represent one polynomial
8064	$n \cdot \lceil \frac{64}{n} \rceil \cdot T(n, k) = \#$ of bits in the public key

Figure 3.1 Bits of Key in Commutative Case (Plaintext = 64 bits)

4 Finding Systems of Practical Size

The public key in the system we have proposed consists of the coefficients of the polynomials making up the transformation P . We do not want a key that is too large and have taken 10,000 bits to be the upper limit on the size of key that we will consider.

4.1 The Commutative Case

We must first count the maximum number of terms in a polynomial of total degree k in n variables. This number, $T(n, k)$ can be computed recursively as follows:

$$\begin{aligned}
 T(1, k) &= k + 1 & (\text{i.e., } 1, X, \dots, X^k) \\
 T(n, 1) &= n + 1 & (\text{i.e., } 1, X_1, \dots, X_n) \\
 T(n, k) &= T(n, k-1) + T(n-1, k) & \forall k, n > 1.
 \end{aligned}$$

	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	
$n=2$	6 384 768	10 640 1280	15 960 1920	21 1344 2688	28 1792 3584	36 2304 4608	45 2880 5760	55 3520 7040	66 4224 8448	78 4992 9984	91 5824 11648
$n=3$	10 430 1290	20 860 2580	35 1505 4515	56 2408 7224	84 3612 10836						
$n=4$	15 480 1920	35 1120 4480	70 2240 8960	126 4032 16128							
$n=5$	21 546 2730	56 1456 7280	126 3276 16380								
$n=6$	28 612 3672	84 1848 11088									
$n=7$	36 684 4788	120 2280 15960									
$n=8$	45 720 5760	165 2640 21120									
$n=9$	55 825 7425	220 3300 29700									
$n=10$	66 858 8580	286 3718 37180									

For multivariate polynomials of degree k in n variables

35	$T(n, k) = \#$ of terms possible in one polynomial
1120	$\lceil \frac{128}{n} \rceil \cdot T(n, k) = \#$ of bits to represent one polynomial
4480	$n \cdot \lceil \frac{128}{n} \rceil \cdot T(n, k) = \#$ of bits in the public key

Figure 4.2 Bits of Key in Commutative Case (Plaintext = 128 bits)

The recursion step follows since $T(n, k-1)$ is the number of terms of total $\leq k$ in which X_n appears. Each such term is of the form X_n times a term in n variables with total degree $\leq k-1$. $T(n-1, k)$ is the number of terms of total degree $\leq k$ in which X_n does not appear.

Now we need the number of bits necessary to represent the coefficients of a polynomial of total degree k in n variables. This clearly depends on the size of the ring but there are restrictions on the ring size if we want the plaintext size to conform to present standards. If the plaintext (X_1, \dots, X_n) is to have 64 bits then each X_i must represent $\lceil \frac{64}{n} \rceil$ bits. If R is a J -ring so that encryption method (3.1) is used, then R must have cardinality $2^{\lceil \frac{64}{n} \rceil}$. The number of bits needed to represent a single polynomial of total degree k in n variables is given by $\lceil \frac{64}{n} \rceil \cdot T(n, k)$. The number of bits in the public key polynomial transformation is $n \cdot \lceil \frac{64}{n} \rceil \cdot T(n, k)$. These numbers are computed and presented in Figure 4.1. The same computation for a plaintext of 128 bits is presented in Figure 4.2.

If the ring is local and the general encryption method of equation 3.3 is used, then the cardinality of the nilpotent ideal, N , must be $2^{\lceil \frac{64}{n} \rceil}$ or $2^{\lceil \frac{128}{n} \rceil}$. The cardinality of R is at least twice that of N . The number of bits needed to represent the public key is at least double the numbers that appear in Figures 4.1 and 4.2. If the specific iteration described in section 2 is used with P_1, \dots, P_n having coefficients in N then the number of bits needed to represent the public key is exactly as shown in Figures 4.1 and 4.2.

It is striking that if we are restricted to 10,000 bits of key then the polynomials making up the encryption transformation and also those making up its inverse can have no more than 153 terms ($n=2, k=16$, 64-bit plaintext). We shall see in the next section that this is too small for cryptographic security.

p	$\log p$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$	$k=14$	$k=15$	$k=16$	$k=17$
2	1	6	10	15	21	28	36	45	55	66	78	91	105	120	136
4	2	12	20	30	42	56	72	90	110	132					
8	3	18	30	45	63	84	108	135							
16	4	24	40	60	84	112	144								
32	5	30	50	75	105	140									
64	6	36	60	90	128										
128	7	42	70	105	147										
256	8	48	80	120											
512	9	54	90	135											
1024	10	60	100												
2048	11	66	110												
4096	12	72	120												
8192	13	78	130												
16384	14	84													
32768	15	90													
65536	16	96													
131072	17	102													
262144	18	108													
524288	19	114													
1048576	20	120													
2097152	21	126													
4194304	22	132													

Figure 4.3 Bits of Plaintext in one Upper-Triangular Matrix over Z_p

4.2 The Upper-Triangular Case

We consider the special case of $k \times k$ upper-triangular matrices over Z_p . We have taken p to be a power of 2 so that there is no bit loss in representing the ring. A single matrix carries $(\log p) \binom{k^2-k}{2}$ bits. In Figure 4.3 we show the number of bits of plaintext in a pair of upper-triangular matrices over Z_p .

Figure 4.3 also provides information in the case of a general commutative ring R . The number of bits shown for a particular p provides lower (upper) bounds when R is any commutative ring with cardinality greater than (less than) p .

p	log p	k	# bits per matrix	$2^k - 1 =$ # terms per polynomial	# bits per polynomial	bits of key = # bits per 2 polynomials
---	-------	---	----------------------	--	--------------------------	--

64 Bit Plaintext

2	1	9	36	511	511	1022
4	2	7	42	127	254	508
8	3	5	45	63	189	378
16	4	5	40	31	124	248
64	6	4	36	15	90	180

128 Bit Plaintext

2	1	12	66	4095	4095	8190
4	2	9	72	511	1022	2044
8	3	8	84	255	765	1530
16	4	7	84	127	508	1016
32	5	6	75	63	315	630
128	7	5	70	31	217	434
2048	11	4	66	15	165	330

Figure 4.4 Bits of Key; Upper-triangular Case

In Figure 4.4 we compute the number of bits in the public key when the plaintext has 64 and 128 bits respectively. To do this computation, we must find the number of terms in a polynomial in 2 upper-triangular $k \times k$ matrices. Recall that these polynomials will have total degree at most $k - 1$. The number of terms of exactly degree j in 2 non-commuting variables is 2^j . It is the same as making j choices from 2 items with repetition allowed. The total number of terms is, therefore:

$$\sum_{j=0}^{k-1} 2^j = 2^k - 1.$$

There appear to be cases where the key is not terribly big and where the number of terms in each polynomial is large enough that we might have cryptographic security. We will see, however, in the next section that we can solve for the coefficients of the polynomials in the inverse transformation in layers so that we need never face a very large system of equations.

	k=2	k=3	k=4	k=5	k=6	k=7
n=2	7 224 448	15 480 960	31 992 1984	63 2016 4032	127 4064 8128	255 8160 16320
n=3	26 572 1718	81 1782 5346	241 5302 15918			
n=4	63 1008 4032	255 4080 16320				
n=5	124 1612 8060	624 8112 40560				
n=6	215 2365 14190					

For multivariate polynomials of degree k in n variables

81
1782
5348

$n^{k+1} - 1 = \# \text{ of terms possible in one polynomial}$
 $\left\lceil \frac{2A}{n} \right\rceil \cdot (n^{k+1} - 1) = \# \text{ of bits to represent one polynomial}$
 $n \cdot \left\lceil \frac{2A}{n} \right\rceil \cdot (n^{k+1} - 1) = \# \text{ of bits to represent the public key.}$

Figure 4.5 Bits of Key; Non-commutative Case; 64-Bit Plaintext

4.3 The General Non-Commutative Case

As will be shown, in the next section, the commutative and upper-triangular cases are not cryptographically secure so we offer one other suggestion. Let R be a non-commutative finite ring. In figure 4.5 we show the number of bits of key in this case. The number of terms in a polynomial of total degree k in n non-commuting variables is given by $n^{k+1} - 1$. The reasoning is the same as that used to compute the number of terms in the polynomials of section 4.1 above. Figure 4.5 shows that there are very few cases to investigate. The number of terms in the polynomials is small but there is some hope that the complications of non-commutative arithmetic will impede cryptanalysis.

5 Inverting These Systems

Assume that we know the public key, $P = (P_1, \dots, P_n)$, and we want to find a transformation $Q = (Q_1, \dots, Q_n)$ such that $Q_i(P_1(\bar{x}), \dots, P_n(\bar{x})) = X_i (i = 1, \dots, n)$. We know that such a system Q of polynomials exists and that the Q_i have the same types of terms as the P_j . That is:

$$Q_i = a_i + b_{i1}V_1 + \dots + b_{in}V_n + c_{i11}V_1^2 + \dots$$

We know which terms are present, we must find the coefficients of $Q_i (i = 1, \dots, n)$.

5.1 If R is a J -ring

Pick a vector $\bar{A} = (A_1, \dots, A_n)$ in R^n . Compute $P_1(\bar{A}), \dots, P_n(\bar{A})$. Set $Q_1(P_1(\bar{A}), \dots, P_n(\bar{A})) = A_1$. This gives a linear equation with coefficients in R whose unknowns are the coefficients of Q_1 . Let q be the number of coefficients of Q_1 . If we can produce vectors $\bar{A}_i = (A_{i1}, \dots, A_{in})$ in $R^n (i = 1, \dots, q)$ such that the resulting linear equations are independent then, hopefully, we can solve for the coefficients of Q_1 . As Q is invertible, we know that such an independent system exists. Any J -ring R is a direct sum of finite fields⁸. Hence the system can be solved independently over each component field of R using standard techniques of linear algebra over fields. A system of approximately 150 equations can be solved in a reasonable time by existing techniques and in our systems of practical size, Q_i never has more than 153 coefficients.

There remains the problem of generating q independent equations. We suggest the following simple procedure. 1. choose a $\bar{A} \not\equiv \bar{0}$ in R_n and accept the linear equations it produces. 2. After

having found $k - 1$ independent equations, choose a new vector $\bar{A} \neq \bar{O}$ at random and accept it if it is independent of the $k - 1$ vectors already found. Otherwise, discard it and repeat this step until you succeed. If, at each stage, the system is put, componentwise, into reduced row-echelon form, then checking the new equation and row-reducing the new system are both easy. We cannot prove that this method will produce the necessary q equations in a reasonable amount of time but believe it does for the following reasons:

Assume $R = Z_p$, p a prime. If k vectors are chosen at random from $(Z_p)^q$ then the probability that they are independent is given by $(1 - \frac{1}{p^q})(1 - \frac{1}{p^{q-1}}) \cdots (1 - \frac{1}{p^{q-k+1}}) > \frac{p-2}{p-1}$ so the probability that a random $q \times q$ determinant over Z_p is non-singular is given by:

$$\prod_{i=1}^q (1 - \frac{1}{p^i}) > \frac{p-2}{p-1}.$$

Although this gives 0 as a lower bound when $P = 2$ the products are actually greater than $1/4$ in this case.

Unfortunately, the coefficient vectors in these equations are not generated at random from all possible p^q vectors over Z_p ; we can only generate p^n vectors but, expect that they will be randomly distributed in the larger set. Given this, the above argument shows that we are likely to generate q independent equations without much difficulty.

5.2 The Nilpotent Case

The message is a vector in N^n where N is a nilpotent ring embedded as a maximal nilpotent ideal in a local ring R . The quotient ring R/N is a finite field. N^n is invariant under the public key polynomial map $P: R^n \rightarrow R^n$. That is, $P: N^n \rightarrow N^n$ is one to one and onto. The component polynomials P_i of P have coefficients in R . To find the coefficients of $Q = P^{-1}$ we first work over the field R/N and then raise the solution to R .

We can assume that P_i ($i = 1 \dots n$) has no constant term. Otherwise $P(X) = T(X) + \bar{C}$ where \bar{C} is a constant vector in N^n and $T(X)$ is a polynomial transformation whose components have no constant terms. P is invertible on N^n if and only if T is. If $U = T^{-1}$ on N^n then: $Q(Y) = U(Y - \bar{C})$ is P^{-1} on N .

Let P_L and Q_L be the linear parts of P and Q . Then

$$\bar{x} = Q(P(\bar{x})) = Q_L(P_L(\bar{x})) + \text{higher order terms.}$$

Let $(P_L \bmod N)$ mean the polynomial obtained by replacing each coefficient, c , of P_L by $(c \bmod N)$. We invert P_L to find Q_L . First find

$$Q_L = (P_L \bmod N)^{-1} \quad \text{over } R/N.$$

Now form Q_L'' with coefficients in R by replacing each coefficient of Q_L' by a representative in R of its class. Then

$$Q_L'' \circ P_L = I + B \quad \text{where } B \text{ has entries in } N$$

and

$$(I - B + B^2 - \dots \pm B^{k-1})Q_L''P_L = I \pm B^k = I \quad \text{on } N^n.$$

So set

$$Q_L = P_L^{-1} = (I - B + B^2 - \dots \pm B^{k-1})Q_L''.$$

Now go on to the quadratic terms. Let P_Q and Q_Q be the quadratic parts and P_H and Q_H the higher order parts of P and Q respectively. Then

$$\begin{aligned}\bar{x} = Q(P(\bar{x})) &= Q_L(P_L(\bar{x}) + P_Q(\bar{x}) + P_H(\bar{x})) + Q_Q(P_L(\bar{x}) + P_Q(\bar{x}) + P_H(\bar{x})) \\ &\quad + Q_H(P_L(\bar{x}) + P_Q(\bar{x}) + P_H(\bar{x})) \\ &= Q_L(P_H(\bar{x})) + Q_L(P_Q(\bar{x})) + Q_Q(P_L(\bar{x})) + \text{higher order terms.}\end{aligned}$$

This gives a system of equations whose unknowns are the coefficients of Q_Q . We can proceed as with the linear parts, finding the coefficients of Q one degree at a time.

For rings of practical size, these systems are therefore too easily solved to be secure.

5.3 The Upper-Triangular Case

The encrypting transformation is $(X_1, X_2) \mapsto P(X_1, X_2) = (P_1(X_1, X_2), P_2(X_1, X_2))$ where X_1, X_2 are upper-triangular matrices over a commutative ring R and P_1, P_2 are polynomials with coefficients in R . To decrypt, we must find a polynomial system $Q = (Q_1, Q_2)$ such that $Q_i(P(X_1, X_2)) = X_i$ ($i = 1, 2$). As before, we can use P and Q to produce pairs (U, V) , $Q(U, V)$ and to set up a system of linear equations in the coefficients of Q . This system is particularly easy to solve as only the linear and constant terms show up in the entries just above the diagonal of $Q(U, V)$. The quadratic terms enter into the entries two levels above the diagonal and so on. We can, therefore, solve for the coefficients of Q in a layered manner, similar to the nilpotent case.

6 Conclusions

We set out to build a public key cryptosystem by repeatedly substituting for variables in multivariate polynomials and simplifying the results to conceal the substitution process. There seems, however, to be no way to build such a system that is both secure and has a public key of practical size when the devices used to limit the number of coefficients are nilpotence and J-rings. We have only shown, however, that it is impossible to produce such a system if the total degree of the encryption polynomial determines the size of the public key. Perhaps, by properly choosing p_0 and p_1 , we can employ the fundamental scheme to produce sparse encrypting polynomials. Then the public key could be kept small while the encrypting polynomial has large total degree and is difficult to invert.

References

- [1] Don Coppersmith and Edna Grossman, "Generators for Certain Alternating Groups with Applications to Cryptography," *SIAM J. Appl. Math.*, Vol. 29, No. 4, pp. 624-627, Dec 1975.
- [2] Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography," *IEEE Trans. Info. Thy.*, Vol. IT-22, No. 6, pp. 644-654, November 1976.
- [3] Data Encryption Standard, FIPS Pub. No. 46, National Bureau of Standards, 15 January 1977.
- [4] Solomon W. Golomb, *Shift Register Sequences*, Holden Day, San Francisco, 1967.
- [5] R. McLeice, A Public-Key Cryptosystem Based On Algebraic Coding Theory, DSN Progress Report 42-44, Jet Propulsion Lab, Calif. Inst. of Tech., Pasadena CA, Jan-Feb 1978.
- [6] R. C. Merkle and M. E. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks," *IEEE Transactions on Information Theory*, Vol. IT-24, No. 5, pp. 525-530, September 1978.
- [7] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *CACM*, Vol. 21, No. 2, pp. 120-126, February 1978.
- [8] Gustavus J. Simmons. Personal Communication.

DEVELOPING AN RSA CHIP

Martin Kochanski
Business Simulations Ltd
Scriventon House
Speldhurst
Kent TN3 0TU
England.

Introduction.

FAP4 is a fast arithmetic processor designed specifically for modular operations, including exponentiation, on large integers. It is at present implemented as an array of 32-bit bit-slice processors, which may be interconnected without additional circuitry to obtain word lengths of up to 1023 bits. With 512-bit operands, exponentiation takes 133 milliseconds at worst (100 ms typically).

Architecture.

FAP4 is based on a new serial/parallel architecture for performing multiplication and modulo reduction together in one pass. It takes one clock cycle per bit of the multiplier, plus a small fixed overhead, to perform a single modular multiplication. While the overall serial one-pass nature of this architecture resembles Brickell's scheme [1], it is less complex, requires less circuitry to implement it, and is less sensitive to the exact details of implementation.

In implementing this architecture, a decision on partitioning had to be made: what should be done in hardware, and what by the host microprocessor?

At one extreme, a design such as Rivest's original one [3] does everything on-chip, even primality testing. The larger the chip, the

more difficult it is to make; and if high-level algorithms are built in to it, what happens if someone develops a better algorithm?

At the other extreme, one could implement as little as possible in hardware. The drawbacks of this can be seen in our own first-ever fast arithmetic processor (FAP1), whose central multiplier chip did a multiplication in 150ns and then waited over 1000ns for the host microprocessor to tell it what to do next!

We therefore decided that the complexity of operations performed by a fast integer arithmetic chip should be such that the host microprocessor knew what to do next by the time the chip had finished: chip time (which was expensive) would not be wasted, and excessive amounts of chip space (also expensive) would not be needed. In concrete terms, this meant that the chip should be able to do at least a single full-length modular multiplication. Even if this took only 30us, it would still leave enough time for the microprocessor to extract (say) the next bit of an exponent and formulate an appropriate command.

Implementation.

Implementing this architecture posed a problem. Typically, one would construct a small prototype using standard TTL chips before planning further designs - but FAP4 involved a "long, thin" design consisting of a few register bits and a little logic at each stage, and such a design does not lend itself to an efficient TTL implementation.

Accordingly, we decided to omit this prototyping phase and implement the architecture directly on semi-custom gate arrays. Because of the serial architecture, it was easy to partition the design into manageable slices, and 32 bits was selected as the slice size. Fujitsu's VH2600 series of 2.3 micron CMOS arrays was chosen.

The chip.

The FAP4 chips are packaged as 64-pin pin grid arrays. Pins are allocated as follows:

Address bus:	10 pins	} Microprocessor interface.
Data bus:	8 pins	
Control lines:	6 pins	
READY signal:	1 pin	
Power:	4 pins	
Clock:	1 pin	

Serial

interconnections: 14 pins (7 between each pair of adjacent chips)

Identifiers: 6 pins (tied to logic 1 or 0 to identify each chip's position in the array)

Internal

control bus: 10 pins

Controller slicing: 3 pins (see below)

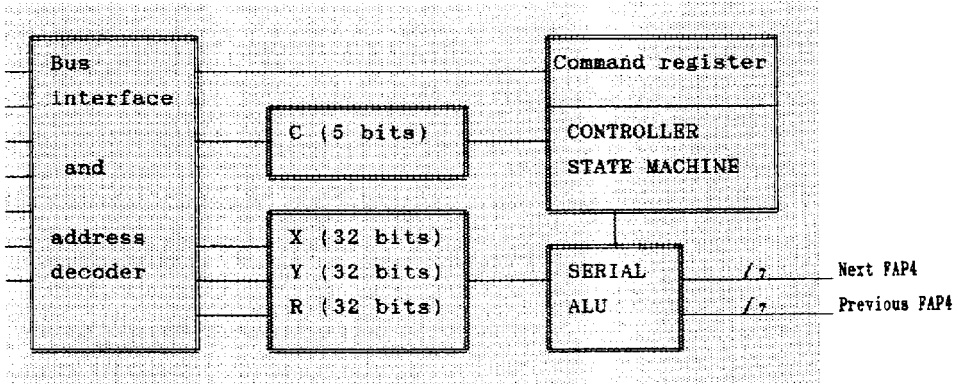
The shaded area shows the pins which exist only because of the partitioning into 32-bit slices. It will be seen that without this overhead, a single-chip implementation would require only 28 pins.

As well as a 32-bit arithmetic slice, each chip contains a 5-bit controller slice. In an array, two of these slices are used together to implement a 10-bit controller for the array, and the remaining controller slices are inactive. This approach allowed the use of a single type of chip throughout, rather than having separate arithmetic and controller chips. An array can thus be expanded to up to 1,024 bits without exceeding the controller's capacity; further expansion requires a separate controller to be provided.

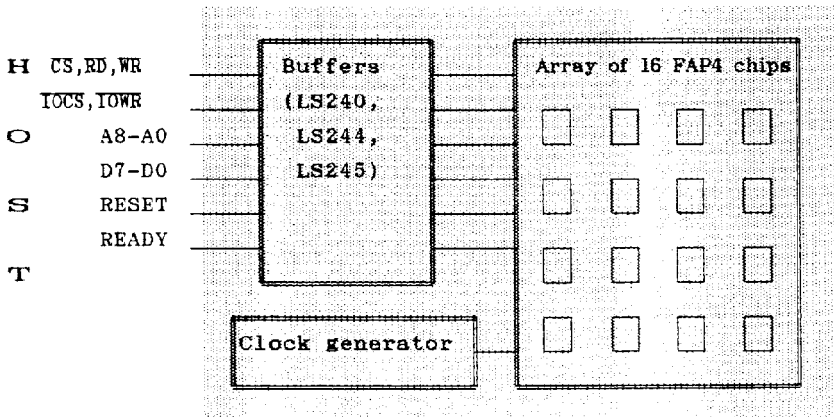
Each chip uses 2,400 2-input NAND equivalent cells, of which about 2,000 are used for the arithmetic slice - about 63 cells per bit.

A typical 512-bit (16-chip) array runs at 5MHz. A 512-bit array will be assumed in all the descriptions of operation and algorithms in this paper.

FAP4 chip block diagram.



FAP4 512 bit array block diagram.



Operation.

The FAP4 array has a standard byte-wide bus interface, and appears to a host microprocessor as a 512-byte memory space and a single output port.

The output port accesses the FAP4 array's command register, and is used to convey commands to the array. As soon as a command is written, the READY interface line goes low and the specified operation is performed. When the operation is completed, READY goes high, and the host can access the array's memory or write a new command to the command register. The READY signal can be interfaced to the host system in a variety of ways: directly to an input port, which the host then polls to test for completion of an operation; to an interrupt controller, so that the host is interrupted whenever the FAP4 array has completed an operation; or to a direct memory access (DMA) controller, so that a sequence of commands can be sent to the FAP4 array without any need for intervention by the host.

Possible commands are as follows (commands are shown in binary, most significant bit first):

00000000	$Y := Y * Y$
00000001	$Y := Y * X$
00000010	$Y := (Y * Y) * X$
00000011	$Y := (Y * X) * X$

where $a * b$ means "multiply register a by the C most significant bits of register b , reducing the result modulo R ".

The first two operations take approximately $C+100$ clock cycles; the last two take twice as long.

Memory is assigned to registers as follows (addresses are in hexadecimal relative to the base of the board's address space):

00 to 7F:	Y register
80 to FF:	X register
100 to 17F:	R register (write-only)
1F8 and 1FC:	C register (write-only)

The X, Y, and R registers are up to 1024 bits (128 bytes) long. In an array of fewer than 32 chips, the X, Y, and R registers are correspondingly shorter, so that in a 16-chip (512-bit) array, register R will extend only from address 140 to 17F.

Register R defines the modulus to be used. If it is zero, then no modulo reduction takes place; otherwise, for correct operation, the two most

significant bits of R must be 01: in other words,

$$2^{510} \leq R \leq 2^{511}-1$$

If the modulus required is outside this range, it can be shifted to fall within it; the Y and X operands can then be shifted to correspond with this.

Registers X and Y define the operand values to be used; the result of each arithmetic operation is stored in register Y , from where it can be read by the host processor. For correct operation, the value of Y should always be less than that of R in modular operations.

Register C is 10 bits long, and is used to define the precision of the current operation, which may be less than the available capacity of the board: smaller values of C produce faster operations.

Operation.

A typical sequence of operations is:

1. Write operands to FAP4's memory.
2. Issue a command.
3. Wait for the READY signal to go high.
4. Read results from FAP4's memory.

Algorithms.

- Multiplication with modulo reduction is the fundamental operation of the chip.
- Multiplication without modulo reduction can be done by setting R to zero.
- Exponentiation is performed by scanning the exponent from left to right and issuing command 10 (square and multiply) if a bit is 1 or 00 (square) if it is 0. These commands can be precomputed and a DMA

channel used to pass them to the array: there is then about a microsecond's latency between commands.

Faster operation could be achieved in an implementation with more registers: for instance, the cube of the base could be stored in a separate register, and the sequence "square, multiply, square, multiply" replaced by "square, square, multiply by cube": but no such extension can do more than double the speed of exponentiation.

- ◆ Division can be performed by multiplication and modulo reduction. A simple example in base 10 is:

$$314159 \times 10000 \text{ modulo } 1469999 = 202137$$

$$\text{so } 314159 / 147 = 2137 \text{ remainder } 20.$$

- ◆ Highest common factor computations can be speeded up by modifying Euclid's algorithm to work with left-justified operands.
- ◆ Modular division (finding y such that $x * y = z \bmod r$) normally uses Euclid's algorithm with a parallel calculation building up the inverse as the algorithm progresses. A similar technique to that used for division can combine the two processes into one.
- ◆ Primality testing is best done by using a procedure such as Knuth's Algorithm P [2], which involves repeated squaring and hence no reloading of operands.

Performance.

At a clock rate of 5MHz, the single multiplication commands 00000000 and 00000001 are executed in $c/5 + 20$ microseconds, where c is the operand length stored in the C register. For full 512-bit operations, this gives a time of 124 μ s per operation. The double-multiplication commands 00000010 and 00000011 take twice as long.

The time taken for a full exponentiation depends on the number of 1 bits in the binary representation of the exponent. The worst-case time for a 511-bit exponentiation is 133ms, and the average time is 100ms.

Applications.

FAP4 can be used for:

- Implementation of the RSA or similar number-theoretic public-key cryptosystems.
- Key setup and exchange for other cryptosystems.
- Authentication of messages and electronic mail by means of "digital signatures".
- Generation of prime numbers and sets of keys for public-key cryptography.
- Add-on acceleration of high-precision fixed-point operations for a host computer.
- Number theoretic research: e.g. evaluation of factorisation algorithms.

Availability.

FAP4 is available now, as chip sets; or on a 512-bit array board with a generic microprocessor interface. Also available are an interface card for the IBM PC; and a special *XBASIC* interpreter, which provides a version of the Basic language which includes support for high-precision arithmetic and for arithmetic operations with built-in modulo reduction.

REFERENCES:

- [1] E.F. Brickell, "A Fast Modular Multiplication Algorithm with Applications to Cryptography", *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, New York: 1983.
- [2] Donald M. Knuth, *The Art of Computer Programming, 2: Seminumerical Algorithms*, Addison-Wesley, New York: 1981 (2nd edition) p.379.
- [3] R.L. Rivest, "A Description of a Single-Chip Implementation of the RSA Cipher", *LAMBDA Magazine* 1, 3 (Fourth Quarter 1980), 14-18.

An M^3 Public-Key Encryption Scheme

H.C. Williams*
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba
CANADA R3T 2N2

1. Introduction. It is well known that the RSA public-key cryptosystem can be broken if the composite modulus can be factored. It is not known, however, whether the problem of breaking any RSA system is equivalent in difficulty to factoring the modulus. In 1979 Rabin [5] introduced a public-key cryptosystem which is as difficult to break as it is to factor a modulus $R = p_1 p_2$, where p_1, p_2 are two distinct large primes. Essentially Rabin suggested that the designer of such a scheme first determine p_1 and p_2 , keep them secret and make R public. Anyone wishing to send a secure message M ($0 < M < R$) to the designer would encrypt M as K , where

$$K \equiv M^2 \pmod{R}$$

and $0 < K < R$, then transmit K to the designer.

The designer can determine M from K by solving the congruences

$$\begin{aligned} x^2 &\equiv K \pmod{p_1} \\ (1.1) \quad y^2 &\equiv K \pmod{p_2} \end{aligned}$$

for x and y . Since $M \equiv \pm x \pmod{p_1}$ and $M \equiv \pm y \pmod{p_2}$, by using the Chinese Remainder Theorem he can deduce four different possibilities for M . If M has some kind of internal redundancy, it should be possible to select the correct M from among the four candidates.

There are two difficulties with this scheme.

- (i) Although there are $O(\log p)$ probabilistic methods for solving the quadratic congruence (see §5)

$$x^2 \equiv M \pmod{p}$$

when p is a prime, the solution of (1.1) and the subsequent use of the Chinese Remainder Theorem can still be quite time consuming.

- (ii) The 4:1 ambiguity in the decrypted messages can be a problem, especially if (as is often the case in transmitting keys) internal redundancy in M is to be minimized.

Indeed, Rabin only advocated his technique as a signature scheme and not as an encryption technique. He also pointed out that, if we insist that $p_1 \equiv p_2 \equiv 1 \pmod{3}$, then we can replace the $K \equiv M^2 \pmod{R}$ step by $K \equiv M^3 \pmod{R}$ and also get a scheme as difficult to break as it is to factor R . However, in this case we get a 9:1 ambiguity in the decrypted messages.

* Research supported by NSERC of Canada Grant A7649.

In [10] Williams showed how a scheme like Rabin's could be developed in which problems (i) and (ii) could be eliminated. This technique made use of the following theorem.

Theorem 1.1 If $p_1 \equiv p_2 \equiv -1 \pmod{4}$, $R = p_1 p_2$, and the Jacobi symbol $(X/R) = 1$. For some X , then

$$X^{(p_1 - 1)(p_2 - 1)/4} \equiv \pm 1 \pmod{R}. \quad \square$$

Corollary. If $K \equiv X^2 \pmod{R}$ and $(X/R) = 1$, then,

$$K^d \equiv \pm X \pmod{R},$$

where $d = ((p_1 - 1)(p_2 - 1)/4 + 1)/2$.

In this scheme the designer determines R and d and a small S such that $(S/R) = -1$. (In [10] R was calculated in such a way that $S = 2$.) He makes R and S public and keeps d secret. Anyone wishing to send a secure message M to the designer

(1) determines b_1 ($= 0$ or 1) such that $(M/R) = (-1)^{b_1}$;

(2) puts

$$M_0 \equiv S^{b_1} M \pmod{R},$$

where $0 < M_0 < R$, and computes b_2 ($= 0$ or 1) such that $b_2 \equiv M_0 \pmod{2}$;

(3) computes

$$(1.2) \quad K \equiv M_0^2 \pmod{R},$$

where $0 < K < R$;

(4) and then transmits $L = \{K, b_1, b_2\}$.

To decrypt L the designer

(1) finds $N \equiv K^d \pmod{R}$,

where $0 < N < R$;

(2) puts $N_0 = R - N$ or N , whichever is even;

(3) and computes

$$M \equiv S^{-b_1} (-1)^{b_1} N_0 \pmod{R}$$

where $0 < M < R$.

This scheme, like Rabin's, is as difficult to break as it is to factor R . Actually, the scheme presented here differs from that given in [10] in two respects. First, it is more general in that it allows for the utilization of an arbitrary S such that $(S/R) = -1$ instead of restricting S to 2. Also in [10] the designer could include a value of e such that $\gcd(e, \phi(R)) = 1$ in his public key $\{R, S, e\}$. This allows for the combination of the above idea with that of the RSA technique. This is easily done by replacing (1.2) above by

$$K \equiv M_0^{2e} \pmod{R}.$$

Of course, the designer must now evaluate his value for d by solving the

linear congruence

$$de \equiv ((p_1 - 1)(p_2 - 1)/4 + 1)/2 \pmod{\phi(R)}.$$

The use of this e values (especially if e is fairly large) will frustrate attacks like those mentioned by Lipton in [1].

The purpose of this paper is to show how this same idea can be extended to the M^3 scheme suggested by Rabin. We first point out that in order to develop our previous cryptosystem it was necessary that we

- I. have the Jacobi symbol and (in order that the scheme be useable) be able to determine the symbol rapidly, i.e. in $O(\log R)$ steps;
- II. have Theorem 1.1;
- III. and have a method for the designer to identify the actual message which was sent (decryption steps (2) and (3)).

Our strategy for extending our idea, then, will be to extend each of I., II., and III.).

2. Arithmetic in $\mathbb{Q}(\rho)$. Let \mathbb{Z} denote the set of all rational integers and let ρ be a primitive cube root of unity, that is $\rho^2 + \rho + 1 = 0$. Let $K = \mathbb{Q}(\rho)$ be the algebraic number field formed by adjoining ρ to the rationals \mathbb{Q} . In this section we will review several of the well-known results concerning K and then develop a theorem analogous to Theorem 1.1.

We first denote by \mathcal{O}_K the set

$$\mathcal{O}_K = \{a + b\rho \mid a, b \in \mathbb{Z}\}.$$

\mathcal{O}_K is the set of all algebraic integers in K . If $\alpha \in \mathcal{O}_K$, then $\alpha = a + b\rho$ for some $a, b \in \mathbb{Z}$ and the norm of α , $N(\alpha)$, is $\alpha\bar{\alpha}$ where $\bar{\alpha} = a + b\rho^2$. Thus $N(\alpha) = a^2 - ab + b^2$.

The primes in \mathcal{O}_K are given by

- (i) $1 - \rho$;
- (ii) p , where p is a prime in \mathbb{Z} and $p \equiv -1 \pmod{3}$;
- (iii) $a + b\rho$, where $a \equiv -1 \pmod{3}$, $3 \nmid b$, and $N(a + b\rho) = p$, where p is a prime in \mathbb{Z} and $p \equiv 1 \pmod{3}$.

Since \mathcal{O}_K is a unique factorization domain, for any $\beta \in \mathcal{O}_K$, we have

$$(2.1) \quad \beta = \gamma \prod_{i=1}^t \pi_i^{k_i},$$

where the π_i ($i = 1, 2, \dots, t$) are primes of \mathcal{O}_K and $\gamma \in \{1, -1, \rho, -\rho\}$. Also, this expression for β is unique (up to order of the π_i 's).

We also have

Theorem 2.1 If $\alpha \in \mathcal{O}_K$ and π is a prime of \mathcal{O}_K , then

$$\alpha^{(N(\pi) - 1)/3} \equiv \rho^\lambda \pmod{\pi},$$

where $\lambda \in \{0, 1, 2\}$. \square

If, with Jacobi, we define the symbol $[\alpha/\pi]$ to be the value of ρ^λ in Theorem 2.1, we can get an extended Jacobi symbol by defining $[\alpha/\beta]$ as

$$[\alpha/\beta] = \prod_{i=1}^t [\alpha/\pi_i]^{k_i},$$

when β has the prime power decomposition given by (2.1).

Let $p_1 \equiv p_2 \equiv 1 \pmod{3}$ be two distinct primes in \mathbb{Z} , $R = p_1 p_2$, and let π_1, π_2 be primes of \mathcal{O}_K such that $N(\pi_1) = p_1$ and $N(\pi_2) = p_2$. Such π_1 and π_2 always exist and in Algorithm 1 of section 5 we describe an expeditious method for finding them. If $\pi_1 = a_1 + b_1 \rho$ and $\pi_2 = a_2 + b_2 \rho$, ($a_1, b_1, a_2, b_2 \in \mathbb{Z}$), then

$$\pi_1 \pi_2 = A + B \rho,$$

where $A = a_1 a_2 - b_1 b_2$, $B = b_1 a_2 - b_2 a_1 - b_1 b_2$ and $\gcd(B, R) = 1$.

Compute $C \in \mathbb{Z}$ by

$$C \equiv -AB^{-1} \pmod{R}.$$

Note that since

$$R = p_1 p_2 = N(\pi_1 \pi_2) = A^2 - AB + B^2,$$

we have

$$C^2 + C + 1 \equiv 0 \pmod{R} \text{ and } C^3 \equiv 1 \pmod{R};$$

indeed,

$$C \equiv \rho \pmod{\pi_1 \pi_2}.$$

We can now prove a result analogous to Theorem 1.1.

Theorem 2.2 If $(p_1 - 1)(p_2 - 1)/9 \equiv -1 \pmod{3}$ and $[X/\pi_1 \pi_2] = 1$ for some $X \in \mathbb{Z}$, then

$$X^{(p_1 - 1)(p_2 - 1)/9} \equiv C^\lambda \pmod{R}$$

where $\lambda \in \{0, 1, 2\}$.

Proof. Let $\rho^\kappa = [X/\pi_1]$ ($\kappa \in \{0, 1, 2\}$).

Since $[X/\pi_1 \pi_2] = 1$, we must have $[X/\pi_2] = \rho^{3 - \kappa}$.

Now $\kappa(p_1 - 1)(p_2 - 1)/9 \equiv -\kappa \pmod{3}$;

hence,

$$(2.2) \quad \kappa(p_1 - 1)/3 \equiv (3 - \kappa)(p_2 - 1)/3 \pmod{3}.$$

We have

$$X^{(p_1 - 1)/3} \equiv \rho^\kappa \pmod{\pi_1}$$

and

$$X^{(p_2 - 1)/3} \equiv \rho^{3 - \kappa} \pmod{\pi_2};$$

thus, if $\lambda \equiv \kappa(p_2 - 1)/3 \pmod{3}$ ($\lambda \in \{0, 1, 2\}$), then from (2.2) we see that

$$x^{(p_1 - 1)(p_2 - 1)/9} \equiv \rho^\lambda \pmod{\pi_1}$$

and

$$x^{(p_1 - 1)(p_2 - 1)/9} \equiv \rho^\lambda \pmod{\pi_2}.$$

It follows that

$$x^{(p_1 - 1)(p_2 - 1)/9} \equiv \rho^\lambda \equiv c^\lambda \pmod{\pi_1 \pi_2}.$$

and

$$x^{(p_1 - 1)(p_2 - 1)/9} \equiv c^\lambda \pmod{R}. \quad \square$$

Corollary. If π_1 and π_2 are defined as above, $K \equiv x^3 \pmod{R}$ and $[x/\pi_1 \pi_2] = 1$, then

$$K^d \equiv c^{\lambda x} \pmod{R},$$

where $\lambda \in \{0, 1, 2\}$ and $d = ((p_1 - 1)(p_2 - 1)/9 + 1)/3$.

3. The M^3 Scheme. In our M^3 scheme the designer selects two large distinct primes p_1, p_2 such that $p_1 \equiv p_2 \equiv 1 \pmod{3}$ and $(p_1 - 1)(p_2 - 1)/9 \equiv -1 \pmod{3}$. He then determines $a_1, a_2, b_1, b_2, A, B, C, d$ as described in §2. He also selects (by trial) a value for $S \in \mathbb{Z}$ such that $[S/\pi_1 \pi_2] = \rho$ and evaluates $S^{-1} \pmod{R}$. He makes his encryption key $\{A, B, S\}$ public. Since $R = A^2 - AB + B^2$, the key occupies the same amount of space as that needed by our M^2 scheme.

To encrypt a message M ($0 < M < R$) the sender executes the following steps.

- (1) Evaluate the extended Jacobi symbol $[M/A + B\rho] = \rho^{b_1}$, where $b_1 \in \{0, 1, 2\}$.
- (2) Determine

$$M_0 \equiv MS^{2b_1}, M_1 \equiv CM_0 \pmod{R},$$

where $0 < M_0, M_1 < R$. Put $M_2 = R - M_0 - M_1$. Since

$M_0 + M_1 + M_2 \equiv R \equiv 1 \pmod{3}$, one of M_0, M_1, M_2 is distinct modulo 3 from the other two. If this is M_i , put $b_2 = i$.

- (3) Compute

$$(3.1) \quad K \equiv M_0^3 \pmod{R},$$

where $0 < K < R$.

- (4) Transmit $E(M) = L = \{K, b_1, b_2\}$.

To decrypt the message L , the designer must perform the following steps.

- (1) Determine

$$N \equiv K^d \pmod{R},$$

where $0 < N < R$.

- (2) Calculate

$$N_0 = N, N_1 \equiv CN_0 \pmod{R} \quad (0 < N_1 < R), N_2 = R - N_1 - N_0.$$

Let N_j be that one of N_0, N_1, N_2 which is distinct modulo 3 from the other two.

(3) Compute

$$D(L) \equiv S^{-b_1} C^{2b_2} N_j \pmod{R},$$

where $0 < D(L) < R$.

That $D(L) = D(E(M)) = M$ follows easily from the corollary of Theorem 2.2 and the simple fact that $C^2 + C + 1 \equiv 0 \pmod{R}$. Hence $\{N_0, N_1, N_2\} = \{M_0, M_1, M_2\}$ and $N_j = M_i \equiv C^i M_0 \pmod{R}$. If, as in the case discussed in §1, we wish to add a value of e such that $\gcd(e, \phi(R)) = 1$ to the encryption key, we can do so easily by replacing (3.1) by

$$K \equiv M_0^{3e} \pmod{R}.$$

Also, d must now be a solution of the linear congruence

$$de \equiv ((p_1 - 1)(p_2 - 1)/9 + 1)/3 \pmod{\phi(R)}.$$

There is, of course, one problem here that we have not discussed and that is the method of computing $[M/A + B\rho]$ rapidly and without knowing how to factor $A + B\rho$. In §5 we describe an $O(\log R)$ algorithm for doing this.

We conclude this section by pointing out that this idea can also be used to produce signatures in much the same manner as that used in [10]; further, our encryption scheme is an example of a claw-free permutation (see Goldwasser et al. [2]).

4. Security. In this section we will show that it is as difficult to break this system as it is to factor R in \mathbb{Z} . This problem is equivalent in difficulty to the problem of factoring $A + B\rho$ in \mathcal{O}_K . We first require three lemmas.

Lemma 4.1. Let $K \equiv Y^3 \pmod{R}$ for some $Y \in \mathbb{Z}$. For any $i \in \{0, 1, 2\}$ there exists an $X \in \mathbb{Z}$ such that

$$X^3 \equiv K \pmod{R} \text{ and } [X/\pi_1 \pi_2] = \rho^i [Y/\pi_1 \pi_2].$$

Proof. Let $j, k \in \{0, 1, 2\}$ such that

$$j - k \equiv i(p_1 - 1)/3 \pmod{3}.$$

Since

$$(p_1 - 1)(p_2 - 1)/9 \equiv -1 \pmod{3},$$

we must have

$$(4.1) \quad i \equiv j(p_1 - 1)/3 + k(p_2 - 1)/3 \pmod{3}.$$

If we use the Chinese Remainder Theorem to find X such that

$$X \equiv C^j Y \pmod{p_1}$$

$$X \equiv C^k Y \pmod{p_2},$$

then

$$X^3 \equiv Y^3 \equiv K \pmod{R}$$

and

$$\begin{aligned} [X/\pi_1 \pi_2] &= [C/\pi_1]^j [C/\pi_2]^k [Y/\pi_1 \pi_2] \\ &= [\rho/\pi_1]^j [\rho/\pi_2]^k [Y/\pi_1 \pi_2] \\ &= \rho^i [Y/\pi_1 \pi_2] \end{aligned}$$

by (4.1).

Lemma 4.2 For any $Y \in \mathbb{Z}$ such that $\gcd(Y, R) = 1$ and any $b_1, b_2 \in \{0, 1, 2\}$ there exists a unique $M \in \mathbb{Z}$ ($0 < M < R$) such that for the encryption key $\{A, B, S, e\}$ we have

$$E(M) = \{K, b_1, b_2\},$$

where $K \equiv Y^3 \pmod{R}$ and $0 < K < R$.

Proof. Let $fe \equiv 1 \pmod{\phi(R)}$
and put $T \equiv Y^{3f} \pmod{R}$.

By Lemma 4.1 there must exist $X \in \mathbb{Z}$ such that $X^3 \equiv T \pmod{R}$ and $[X/\pi_1 \pi_2] = 1$. Define $X_i \equiv C^i X \pmod{R}$, where $0 < X_i < R$, $i = 0, 1, 2$, and let X_j be that one of X_0, X_1, X_2 which is distinct modulo 3 from the other two. Set

$$k \equiv 2(b_2 - j) \pmod{3}, \quad k \in \{0, 1, 2\}$$

and put

$$M \equiv S^{-2b_1} C^k X \pmod{R},$$

where $0 < M < R$.

Now

$$\begin{aligned} [M/\pi_1 \pi_2] &= [S/\pi_1 \pi_2]^{-2b_1} = \rho^{b_1} \\ ([C/\pi_1 \pi_2] &= \rho^{(p_1 - 1) + (p_2 - 1)/3} = 1) \end{aligned}$$

also,

$$M_i \equiv C^i S^{-2b_1} M \equiv C^h X \pmod{R},$$

where $h = 2(b_2 - j) + i$ and $0 < M_i < R$. Hence, we get $\{M_0, M_1, M_2\} = \{X_0, X_1, X_2\}$ and when $i = b_2$, then

$$M_i \equiv C^j X \pmod{R}.$$

It follows that $M_i = X_j$ and M_i is distinct modulo 3 from the other two M_m values when $i = b_2$. Also

$$M_0^{3e} \equiv X^{3e} \equiv T^e \equiv Y^{3ef} \equiv Y^3 \equiv K \pmod{R}.$$

Hence $E(M) = \{K, b_1, b_2\}$. Since $D(E(M)) = M$, M must also be unique. \square

Lemma 4.3 If $X, Y \in \mathbb{Z}$, $X^3 \equiv Y^3 \pmod{R}$, and $[X/\pi_1 \pi_2] \neq [Y/\pi_1 \pi_2]$, then $\gcd(X - C^i Y, R) = p_1$ for some $i \in \{0, 1, 2\}$.

Proof. Since $X^3 \equiv Y^3 \pmod{R}$, we have

$$(X - Y)(X - CY)(X - C^2 Y) \equiv 0 \pmod{p_1 p_2}.$$

If $p_1 p_2 \mid X - C^i Y$, then

$$[X/\pi_1 \pi_2] = [C^i Y/\pi_1 \pi_2]_2 = [Y/\pi_1 \pi_2],$$

which is not so. Thus, there must exist some $X - C^i Y$ with $i \in \{0, 1, 2\}$ such that $p_1 \mid X - C^i Y$ and $p_2 \nmid X - C^i Y$. It follows that $\gcd(X - C^i Y, R) = p_1$. \square

Now suppose that we have some algorithm F which we will decrypt $1/k$ of all messages. If an arbitrary Y is selected such that $[Y/\pi_1 \pi_2] \neq 1$ and $\gcd(Y, R) = 1$ (Note that S is a possible value of Y), then put $K \equiv Y^3 \pmod{R}$

with $0 < K < R$ and select any $b_1, b_2 \in \{0, 1, 2\}$. By Lemma 4.2 there exists a unique M such that

$$E(M) = \{K, b_1, b_2\}.$$

After k trials at a value for Y we would expect that F would determine the corresponding M from $\{K, b_1, b_2\}$. Putting $M_0 \equiv MS^{2b_1} \pmod{R}$ and $X \equiv M_0^e \pmod{R}$, we have

$$X^3 \equiv Y^3 \pmod{R}$$

and

$$1 = [X/\pi_1 \pi_2] \neq [Y/\pi_1 \pi_2].$$

It follows from Lemma 4.3 that with knowledge of M and Y , we can easily factor R .

It might be felt that, in revealing the values of A and B , the designer in some way aids his opponent to factor $R = A^2 - AB + B^2$. For example, if his opponent were able to find G, H such that $G \neq \pm A, \pm B, \pm(A - B)$ and $R = G^2 - GH + H^2$, then he could factor R by using his knowledge of A, B, G, H . We point out, however, that if we are given C such that $C^2 + C + 1 \equiv 0 \pmod{R}$, then $(2C + 1)^2 \equiv -3 \pmod{R}$ and it can be shown that by using Algorithm 1 we can compute A and B such that

$$R = A^2 + AB + B^2$$

in $O(\log R)$ operations. Thus, knowledge of C is equivalent to the knowledge of A and B . Now $C^3 \equiv 1 \pmod{R}$ and if we could find X such that $X^3 \equiv 1 \pmod{R}$ and $[X/\pi_1 \pi_2] \neq 1$, we could factor R . But this is really no different from taking an arbitrary Y , determining $K \equiv Y^3 \pmod{R}$ and then finding some X such that $X^3 \equiv K$ and $[X/\pi_1 \pi_2] \neq [Y/\pi_1 \pi_2]$, a problem equivalent in difficulty to factoring R . That is, unless there is something special about a value of $K = 1$, knowledge of C seems, for the problem of factoring R , to give no more information than the knowledge of an arbitrary Y .

We should, nevertheless, emphasize here that the method of showing the equivalence of breaking our system to the problem of factoring R is constructive; that is, this encryption technique is vulnerable to a known cipher text attack, if such an attack can be mounted. We refer the reader to the relevant comments in [10] concerning this.

The problem of extending our method further to an M^r encryption scheme, where r is a prime and $p_1 \equiv p_2 \equiv 1 \pmod{r}$ is rather difficult. In the first place, it is necessary to be able to further extend the Jacobi symbol and be able to evaluate it in $O(\log R)$ time. This would mean, as far as is known today, that the cyclotomic extension of the rationals $K_r = \mathbb{Q}(\rho)$, where ρ is a primitive r^{th} root of unity, must be Euclidean. As K_r can be Euclidean only when the class number of K_r is 1, this means that r could only be 2, 3, 5, 7, 11, 13, 17, 19. Of these it is known that if $r = 2, 3, 5, 7, 11$, then K_r is Euclidean. The other values 13, 17, 19 have not

been investigated (see Lenstra [4]). While it may, in principle, be possible to extend the algorithms in §5 to the cases of $r = 5, 7, 11$, the details would be very onerous and the corresponding computations would be concomitantly slowed. Possibly, the case of $r = 5$ might be worthwhile investigating.

5. Algorithms. In this section we describe two algorithms. The first of these is a method of determining a and b , given m and x such that $x^2 \equiv -3 \pmod{m}$, for which

$$m = a^2 - ab + b^2.$$

If m is a prime we can find x in $O(\log m)$ operations by using either the algorithm described by Lehmer [3] or that of Shanks [7]. This often requires that we know in advance a quadratic non-residue of m . There is no $O(\log m)$ deterministic way known for doing this, but in practice one finds such a non-residue by trial very easily. An $O(\log m)$ deterministic method for finding x when m is prime has been given recently by Schoof [6], but as Schoof himself says, no one would ever use this very complicated technique.

The algorithm we present here is a simple adaptation of the method described by Wilker [8] to solve $u^2 + 5v^2 = m$. There is no loss of generality in assuming m is not a perfect square and $m \equiv 1 \pmod{3}$.

Algorithm 1. (Find s, t such that $m = s^2 + 3t^2$ when $m \equiv 1 \pmod{3}$.)

- (1) Use the Euclidean algorithm to find r_0, r_1, r_2, \dots , where

$$\begin{aligned} x &= q_0 m + r_0 & 0 < r_0 < m \\ m &= q_1 r_0 + r_1 & 0 < r_1 < r_0 \\ r_0 &= q_2 r_1 + r_2 & 0 < r_2 < r_1 \\ &\vdots & \vdots \end{aligned}$$

If $r_0^2 < m$, then $m = r_0^2 + 3$ and we are done. If $r_0^2 > m$, then find r_n such that

$$r_n^2 - 1 > m \text{ and } r_n^2 < m.$$

Only $O(\log m)$ operations are needed to do this.

- (2) Put $s = \pm r_n$. When $3 \nmid r_n - 1$ and $r_n^2 - 1 < 9m$, put $t = +r_n - 1/3$; otherwise, put $t = \pm(r_n - k)$, where

$$k \equiv ((3r_n \varepsilon_n - 1 - \varepsilon_n \varepsilon_n - 1)r_n - 2r_n - 1)/6 \pmod{r_n}.$$

Here $0 < k < r_n$, $r_i \equiv \varepsilon_i \pmod{3}$, and $|\varepsilon_i| \leq 1$.

$$\text{We have } m = s^2 + 3t^2 = (s + t)^2 - 2t(s + t) + 4t^2.$$

If m is a prime p and we want a prime $\pi = a + bp$ such that $N(\pi) = p$, then we select the sign of s such that $a = s + t \equiv -1 \pmod{3}$ and put $b = -2t$ when $3 \nmid t$. If $3 \nmid t$, we select the sign of t such that $a = 2t \equiv -1 \pmod{3}$ and put $b = s + t$.

The next algorithm we present is one which can be used to evaluate the extended Jacobi symbol $[\alpha/\beta]$ without requiring the factorization of β . This algorithm was undoubtedly known to Jacobi and is given in Williams and Holte [9]. We assume that $\alpha = A + B\rho$, $\beta = C + D\rho$. Here the symbols A, B, C, D do not have the meanings assigned to them previously but merely denote rational integers such that $3 \nmid D$ and $3 \mid D$.

Algorithm 2. (Determine g and γ such that $[\alpha/\beta] = \rho^g[\beta/\gamma]$ and $N(\gamma) < N(\beta)$).

- (1) Find $E = A - xC + yD$, $F = B - yC - xD + yD$, where

$$x = \text{Ne}\{(AC + BD - AD)/N(\beta)\},$$

$$y = \text{Ne}\{(BC - AD)/N(\beta)\},$$

$$N(\beta) = C^2 - CD + D^2, \quad \text{Ne}\{\alpha\} \text{ denotes the nearest integer to } \alpha.$$

- (2) If $E \equiv -F \pmod{3}$, divide $E + F\rho$ by $1 - \rho$ k times until $(E + F\rho)/(1 - \rho)^k = \bar{E} + \bar{F}\rho$ and $\bar{E} \not\equiv -\bar{F} \pmod{3}$. This process is facilitated by making use of the observation that if $E = -F + 30$, then

$$(E + F\rho)/(1 - \rho) = 2Q - F + Q\rho.$$

- (3) If $3 \mid \bar{F}$, put $j = 0$, $G = \bar{E}$, $H = \bar{F}$; if $3 \mid \bar{E}$, put $j = 1$, $G = \bar{F} - \bar{E}$, $H = -\bar{E}$; if $3 \nmid \bar{F}\bar{E}$, put $j = 2$, $G = -\bar{F}$, $H = \bar{E} - \bar{F}$. Then $\gamma = G + H\rho$ and $g \equiv (2k + j)(C^2 - 1)/3 - jCD/3 \pmod{3}$.

We have $[\alpha/\beta] = \rho^g[\beta/\gamma]$ and $N(\gamma) < 3/4 N(\beta)$. Clearly we can repeat this algorithm until we get a symbol of the form $[\pm 1/\lambda] = 1$; the accumulated power of ρ will then be the value of $[\alpha/\beta]$. Since $N(\gamma) < 3/4 N(\beta)$, we see that this algorithm must terminate in $O(\log N(\beta))$ operations.

REFERENCES

- [1] R.A. Demillo, G.I. Davida, D.P. Dobkin, M.A. Harrison, and R.J. Lipton, **On the Safety of Cryptosystems**, Applied Cryptology, Cryptographic Protocols and Computer Security Models, AMS Short Courses Lecture Notes, Vol. 29, Providence, 1983.
- [2] Shafi Goldwasser, Silvio Micali, R.L. Rivest, A **"paradoxical" solution to the signature problem**, Proc. 25th IEEE Symposium on Foundations of Computer Science, to appear.
- [3] D.H. Lehmer, **Computer technology applied to the theory of numbers**, Studies in Number Theory, Math. Assoc. of America, 1969, Theorem 5, p. 133.
- [4] H.W. Lenstra, jr., **Euclidean number fields I.**, Math. Intelligencer 2 (1979/80), 6 - 15.
- [5] M.O. Rabin, **Digitized signatures and public-key functions as intractable as factorization**, M.I.T. Lab. for Computer Science, Tech. Rep. LCS/TR212, 1979.
- [6] Rene Schoof, **Elliptic curves over finite fields and the computation of square roots mod p**, Math. Comp. 44 (1985), 483 - 494.
- [7] D. Shanks, **Five number theoretic algorithms**, Congressus Numerantium 7 (1973), 51 - 69.
- [8] Peter Wilker, **An efficient algorithmic solution of the diophantine equation $u^2 + 5v^2 = m$** , Math. Comp. 35 (1980), 1347 - 1352.
- [9] H.C. Williams and R. Holte, **Computation of the solution of $x^3 + Dy^3 = 1$** , Math. Comp. 31 (1977), 778 - 785.
- [10] H.C. Williams, **A modification of the RSA public-key encryption procedure**, IEEE Transactions on Information Theory, IT-26 (1980), 726 - 729.

Trapdoor Rings And Their Use In Cryptography.

V.Varadharajan

Dept. of Elec. and Electronic Eng.,

Plymouth Polytechnic,

Drake Circus, PLYMOUTH PL4 8AA,

U.K.

Abstract

This paper examines possible trapdoor structures which can be used to design public key cryptosystems based on the factorization problem. Some examples of such finite trapdoor systems which might serve as a basis for an extended RSA cryptosystem are proposed.

Introduction

Recently much research work has been carried out in the field of asymmetric or public key cryptosystems [1,2,3], which allow two users to communicate securely over an insecure channel without any prearrangement. They are classified as asymmetric because the sender and the receiver employ two different keys to encrypt and decrypt a message. Separating the enciphering and deciphering capabilities allows secrecy to be maintained without keeping the encrypting key hidden as it is no longer used in deciphering. The decrypting key is kept private and there is no need for anyone to communicate his decryption key to anyone else. The concept of a public key cryptosystem is illustrated in figure 1. User 1

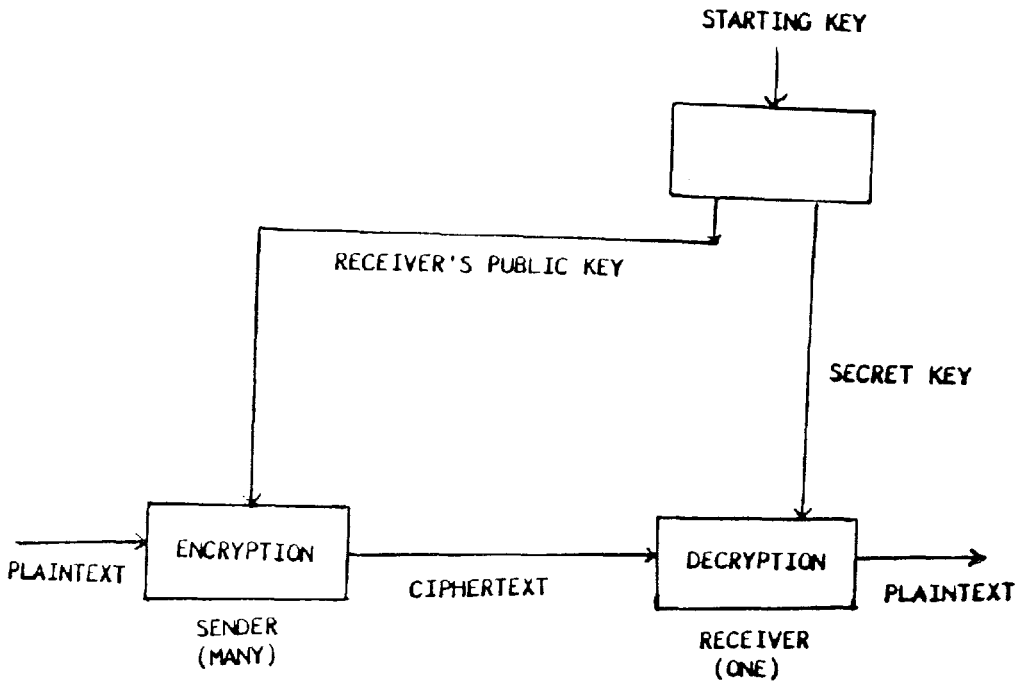


Fig.1 - Public key cryptosystem

encrypts the message M using the publicly known encrypting key of user j and sends the cipher to user j over an insecure channel. Only the user j will be able to decrypt the cipher to recover M as he is the only one who knows his secret decryption key.

The encryption (E) and decryption (D) algorithms in such a system have, in general, the following properties:

- Deciphering the enciphered form of a message M yields M , that is, $D(E(M)) = M$.
- Both E and D are easy to compute.
- By publicly revealing E , the user does not reveal an easy way to compute D . This means that only the receiver can decrypt the messages encrypted with E or compute D efficiently.

A major implication of the public key cryptosystem is that it eliminates the need for a secret transferral of keys as in the case of conventional symmetric cryptographic algorithms which employ the same key for both encryption and decryption. Furthermore, the public key algorithms can be used in conjunction with the symmetric algorithms to distribute the secret key. This can be seen as follows:

User i can encrypt the secret key in a symmetric system using the public key of user j and then send it to user j over an insecure channel. Because the deciphering key is only known to user j, he is the only one who can decrypt the cipher and obtain the secret key. Now users i and j can have a secure conversation using a symmetric algorithm with the transferred secret key.

Note that in this arrangement the sole purpose of the public key system is to distribute the secret key required for the symmetric algorithm.

Another implication of the public key cryptosystem is that it is possible to 'sign' messages in a way that is unforgeable but easily verifiable. This can be accomplished provided the enciphering and deciphering procedures can be used in either order. To sign a message, a user i operates first on the message with his secret decryption key and then with the public key of user j to produce the cipher. The user j recovers the message by operating on the cipher, first with his secret decryption key and then with the public key of user i. Since only user i knows his secret decryption key, only user i could have created the cipher which produces the correct message when his public key is applied to it. Thus it is possible to obtain a digital signature feature [3] provided the encryption and decryption algorithms satisfy an additional property (d) given by

d. Enciphering the deciphered form of a message M yields M , that is, $E(D(M))=M$.

For a public key system to be secure, it should be computationally infeasible for the cryptanalyst to determine the secret decryption key from the publicly known parameters of the encryption and decryption procedures and the encryption key. Such systems are constructed using 'trapdoor one-way functions'

Definition: A function f is said to be a one-way function if it is easy to compute $y=f(x)$ for all x but difficult to compute $x=f^{-1}(y)$ for almost all y .

Note that the phrase 'almost all y ' is necessary because a table of some of the values of $f(x)$ can be stored and if y happens to belong to this table the corresponding x can be easily determined. The above definition does not provide an absolute sense in which a function is one-way as it depends on the computational resources available. A precise definition of a one-way function depends on a specific measure of complexity as the difficulty of computing the inverse function varies with time and technology. The complexity measures are often defined in terms of time or storage required to compute the inverse. Computational tasks which require of the order of 10^{50} operations or 10^{50} storage elements are generally considered to be infeasible [4]. It is possible to construct one-way functions $y=f(x,k)$ where the difficulty of computing y increases linearly with k but that of computing x increases exponentially with k . In such a case it is possible to increase k to such an extent that computation of the inverse requires the limits mentioned above. However one-way functions cannot be used directly to design a public key system as the legal receiver needs to decrypt the cipher y easily for all y .

Definition: A 'trapdoor one-way function' is a one-way function with the additional property that if certain specific information (the trapdoor) employed in the design of the function is known then it is easy to compute the inverse function.

That is, given the secret decryption key (trapdoor) $f^{-1}(y)$ can be easily calculated.

A well known public key cryptosystem which has survived many cryptanalytical attacks is the Rivest-Shamir-Adleman system (RSA) [2], which is based on the difficulty of factoring a large rational integer into its primes. The system designer chooses two distinct primes p and q and publishes the product $m=pq$. The product m is assumed to be so large that factoring it is beyond all projected computation capabilities. For instance, if m is chosen to be 200 digit decimal integer, then it will require of the order of 10^{23} operations using the best known factoring algorithm [2]. The encryption procedure raises the message $x, 1 < x < m$, to the e -th power modulo m and the decryption is performed by raising the cipher y to the d -th power modulo m .

Encryption : $y \equiv x^e \pmod{m}$

Decryption : $x \equiv y^d \pmod{m}$

The public encryption key is (e, m) and the secret decryption key is (d, m) . The coding exponents e and d are chosen to be multiplicative inverse of each other modulo $\phi(m)$, where ϕ is the Euler totient function.

$$ed \equiv 1 \pmod{\phi(m)} \quad (1)$$

If the cryptanalyst can determine $\phi(m)$, then he can obtain the decoding exponent d by solving (1). If m can be easily factorized to p and q , then the cryptanalyst can find $\phi(m)$ and d and hence crack the system.

In this article, some algebraic structures suitable for use in the design of RSA type factorization-based, trapdoor systems are investigated.

Trapdoor Rings

Assume R is a finite ring with unity which is associative but not necessarily commutative. Suppose that members of the ring R are used as messages and that $r \in R$ is enciphered as r^e where e is the published encrypting exponent.

The trapdoor property can be stated as follows:

there exists some integer $n > 0$ such that $r^{n+1} = r$ for all $r \in R$.

These rings are to be referred to as trapdoor rings.

For instance, in the ring of integers modulo a prime number p , $R = \mathbb{Z}/p\mathbb{Z}$, $r^p = r$ for all $r \in R$. More generally, if $R = F_q = GF(q)$, the field of q elements where q is a prime power (p^k), then $r^q = r$ for all $r \in R$.

Consider any two such trapdoor rings R and S and construct the direct sum $R \oplus S$ consisting of vectors (r, s) with $r \in R$ and $s \in S$. These vectors are added and multiplied componentwise. That is, $(r, s) + (r', s')$ gives $(r+r', s+s')$ and $(r, s) \cdot (r', s')$ is (rr', ss') . These rules make $R \oplus S$ into another trapdoor ring, say T . The number of elements of T is equal to the product of the number of elements in rings R and S . Suppose that $r^{m+1} = r$ for all $r \in R$ and $s^{n+1} = s$ for all $s \in S$. Then $r^{m+N} = r^N$ for all $N \geq 1$. Similarly, $s^{n+N} = s^N$ for all $N \geq 1$. In particular, $r^{1+km} = r$ for all $k \geq 1$ and $s^{1+kn} = s$ for all $k \geq 1$. Hence $r^{1+lmn} = r$ and $s^{1+lmn} = s$ for all $l \geq 1$. Hence $t^{1+lmn} = t$ for all $t \in T$, $l \geq 1$, and so T is a trapdoor ring.

This above process can be applied repeatedly by taking vectors of

arbitrarily many components, each taken from some finite field. Considering finite fields F_{q_i} for $1 \leq i \leq j$ where q_i 's can be the same or different, the trapdoor ring R consists of all vectors $x = (x_1, \dots, x_j)$ where $x_i \in F_{q_i}$, $1 \leq i \leq j$. The ring consists of $q_1 \dots q_j$ elements and the equality $r^{n+1} = r$ is obeyed for all $r \in R$, where n is equal to $(q_1-1) \dots (q_j-1)$ or any multiple of it.

There are many finite rings which are not trapdoor rings. Consider, for instance, $R = \mathbb{Z}/p^2\mathbb{Z}$ where p is a prime. Then $p^2 = 0 = p^3 = \dots$ in ring R but $p \neq 0$ in R . So the property that $p^{n+1} = p$ is not satisfied for any $n > 0$. However if we take an integer k to be a square free positive integer, say, $k = p_1 \dots p_j$ where all p_i 's are distinct primes then the ring $R = \mathbb{Z}/k\mathbb{Z}$ is a trapdoor ring and in fact it can be regarded as a direct sum of $F_{p_1} \oplus \dots \oplus F_{p_j}$ as described above. If $j = 2$, then this becomes the standard trapdoor ring used by the RSA cryptosystem.

Classification Theorem

Let R be any trapdoor ring. Then

- a. R has a 1.
- b. R is commutative.
- c. R is isomorphic to $F_{q_1} \oplus \dots \oplus F_{q_j}$ for certain finite fields.

(Two rings R and S are isomorphic to each other if there exists a function $f : R \rightarrow S$ which is one-to-one and onto and satisfies $f(r_1 + r_2) = f(r_1) + f(r_2)$, $f(r_1 r_2) = f(r_1) f(r_2)$ for all $r_1, r_2 \in R$.)

The proof relies on the use of Wedderburn's structure theory [5] for semisimple rings. The main steps of the argument are as follows :

1. The ring R is trapdoor implies that R has no nilpotent elements except 0. (An element x is said to be nilpotent if $x^a = 0$ and $x^{a-1} \neq 0$ for some $a > 0$)
2. A finite ring without non-zero nilpotent elements must have a 1 [5].
3. A finite ring with 1 and lacking nilpotents ($\neq 0$) is a direct sum of matrix rings with entries in a division algebra (skew field) - Wedderburn's theorem.
4. If any of these matrices is not 1×1 , then there will be non-zero nilpotent elements in R .
5. Hence R is a direct sum of finite skew fields.
6. A finite skew field is necessarily commutative (Wedderburn-Witt theorem).

Hence the only finite trapdoor rings are of the type described above upto isomorphism.

Other possible trapdoor structures

Instead of using rings, we only need a system S in which an associative multiplication is defined, satisfying

1. For all $a, b \in S$, $ab \in S$
2. For all $a, b, c \in S$, $a(bc) = (ab)c$

Such a system is called a semigroup. If S is a semigroup with a 1 satisfying $a \cdot 1 = 1 \cdot a = a$ for all $a \in S$ then S is said to be a monoid.

Further if a monoid S has the additional property that there is a unique corresponding $b \in S$ such that $ab = ba = 1$ then S is said to be a group. In a finite group G with n elements, every $g \in G$ satisfies $g^n = 1$, $g^{n+1} = g$.

Hence a finite group of order n can be used to construct trapdoor

systems. On the other hand, not all semigroups can be used to form trapdoor systems. Semigroups with the property that $a^{n+1} = a$ for all $a \in S$ are possible candidates. A more formal way of expressing this constraint would be : a semigroup S can be used provided it is completely regular.

Let us now consider some examples of finite systems that might serve as a basis for a generalized RSA cryptosystem.

Ring of Matrices

If the ring of all $n \times n$ matrices M_n over the ring $R = \mathbb{Z}/m\mathbb{Z}$, where

$m = \prod_{i=1}^s p_i^{r_i}$, (p_i 's are primes), is considered, then the ring M_n

contains nilpotent elements when $n > 1$. This problem can be overcome by

restricting the message space thereby avoiding nilpotent elements. In

this paper, we consider three such subsets namely, the set of non-singular matrices over $\mathbb{Z}/m\mathbb{Z}$, the set of upper triangular matrices over $\mathbb{Z}/m\mathbb{Z}$ and the set of orthogonal matrices over $\mathbb{Z}/m\mathbb{Z}$.

Let us first consider the multiplicative group formed by the non-singular matrices of order n over $\mathbb{Z}/m\mathbb{Z}$. The order of the group, N_m , is given by

$$N_m = \prod_{i=1}^s N_{p_i^{r_i}} \quad (2)$$

where $N_{p_i^{r_i}}$ denotes the order of the group formed by non-singular matrices over $\mathbb{Z}/p_i^{r_i}\mathbb{Z}$.

It is well known [6] that the order of the group formed by non-singular matrices over $\mathbb{Z}/p\mathbb{Z}$ is given by

$$N_p = (p^n - 1)(p^n - p) \dots (p^n - p^{n-1}) \quad (3)$$

To evaluate N_{p^r} , in general, let Θ be the homomorphism mapping an $n \times n$ matrix A over $\mathbb{Z}/p^{r+1}\mathbb{Z}$ to A' , a matrix over $\mathbb{Z}/p^r\mathbb{Z}$, via $a_{ij} \pmod{p^{r+1}} \longrightarrow a_{ij} \pmod{p^r}$. This induces a surjective homomorphism Θ' between the linear groups formed by these matrices. That is,

$$\Theta' : GL_n(\mathbb{Z}/p^{r+1}\mathbb{Z}) \longrightarrow GL_n(\mathbb{Z}/p^r\mathbb{Z})$$

Using group theory [7],

$$\frac{GL_n(\mathbb{Z}/p^{r+1}\mathbb{Z})}{\text{Kernel } \Theta'} \cong GL_n(\mathbb{Z}/p^r\mathbb{Z})$$

where \cong denotes isomorphic to.

The kernel consists of the set of matrices which are mapped to the identity matrix $I \pmod{p^r}$, i.e.

$$a_{ii} \equiv 1 \pmod{p^r} \quad \text{for } 1 \leq i \leq n \quad (4)$$

$$a_{ij} \equiv 0 \pmod{p^r} \quad \text{for } i \neq j \quad (5)$$

There are p possibilities for each of the equations (4) and (5) giving rise to p^{n^2} possibilities. Therefore using group theory, the order (denoted by symbol $\#$) is given by

$$\begin{aligned} \# GL_n(\mathbb{Z}/p^{r+1}\mathbb{Z}) &= p^{n^2} \# GL_n(\mathbb{Z}/p^r\mathbb{Z}) \\ &= p^{rn^2} \# GL_n(\mathbb{Z}/p\mathbb{Z}) \\ &= p^{rn^2} N_p \end{aligned}$$

Thus using (3),

$$N_{p_i}^{r_i} = p_i^{(r_i-1)n^2} (p_i^{n-1}) \dots (p_i^n - p_i^{n-1}) \quad (6)$$

Substituting (6) into (2) gives the order N_m .

Now as in the RSA cryptosystem, if m is made to be the product of two distinct primes p and q , then the expression for N_m simplifies to

$$\begin{aligned} N_m &= N_p \ N_q \\ &= (p^{n-1}) \dots (p^n - p^{n-1}) (q^{n-1}) \dots (q^n - q^{n-1}) \end{aligned}$$

A public key system can therefore be constructed using (e, m, n) as the public encryption key and (d, m, n) as the secret decryption key. The coding exponents e and d are determined using

$$ed \equiv 1 \pmod{N_m} \quad (7)$$

A message $M \in GL_n(Z/mZ)$ obeys

$$M^{N_m} \equiv I \pmod{m}$$

The encryption and decryption procedures can therefore be given by

$$C \equiv M^e \pmod{m}$$

and

$$M \equiv C^d \pmod{m}$$

respectively, where $M, C \in GL_n(Z/mZ)$.

Although the order N_m can be used in finding e and d as in (7), in practice, it is often desirable to find the exponent, EXP , of the group, that is, the least integer greater than zero such that

$$M^{EXP} \equiv I \pmod{m} \quad M \in GL_n(Z/mZ)$$

The exponent of the group can be shown to be [8,9].

$$\text{EXP} = \text{lcm} \{v_1, \dots, v_s\}$$

where

$$v_i = p_i^{r_i-1} (w_i)$$

$$\text{and } w_i = p_i \text{ lcm} (p_i-1, p_i^2-1, \dots, p_i^{n-1})$$

(8)

(assuming p_i is greater than n for all i).

As the expression for the order N_m (and the exponent EXP) depend on the prime factors of m , it can be used to design a public key cryptosystem by choosing m to be a large integer.

Alternatively, let us now consider the set of upper triangular matrices as a possible choice for the message space. If the diagonal entries are made unity, to ensure that the matrix is invertible over any modulus, then the order of the group formed by such matrices over $\mathbb{Z}/m\mathbb{Z}$ is equal to $\text{ord} = m^{n(n-1)/2}$. That is, the order does not depend on the prime factors of m and hence this cannot be used as a public key system. A conventional cryptosystem can be designed where the secret key is (e, d, m, n) and the exponents e and d are calculated using $ed \equiv 1 \pmod{\text{order}}$.

However, if the message space is altered to contain upper triangular matrices with diagonal entries relatively prime to m , then such matrices are again invertible modulo m . Further, in practice, as m is a product of large prime numbers, the choice of diagonal elements is almost arbitrary provided they are chosen to be relatively small integers.

The order of the group formed by such matrices is determined as follows:-

Considering a $n \times n$ matrix, it is required that all the n diagonal entries must be coprime to m . The number of integers less than m and coprime to m is given by the Euler totient function $\phi(m)$. The remaining $n(n-1)/2$ superdiagonal entries of the matrix may take any value modulo m . Therefore, the order is equal to $m^{n(n-1)/2} \{\phi(m)\}^n$. The vital difference between this order and the one calculated above is that now the order of the group is dependent on the prime factors of m . Hence the modulus m needs to be factorized before the decryption exponent d can be calculated using $ed \equiv 1 \pmod{\text{order}}$. As for the set of non-singular matrices, the exponent of the group formed by such upper triangular matrices can be used instead of the order in finding e and d . The exponent of the group is shown to be equal to [8,9].

$$\text{EXP}' = \text{lcm} \{ \phi(p_1^{r_1}) p_1^{r_1}, \dots, \phi(p_s^{r_s}) p_s^{r_s} \}$$

where

$$m = \prod_{i=1}^s p_i^{r_i}$$

Finally, one can also use the special set of non-singular matrices, namely the set of orthogonal matrices, as the message space of the matrix based RSA system. The order of the group formed by $n \times n$ orthogonal matrices over $\mathbb{Z}/p\mathbb{Z}$ has been worked out by MacWilliams [10].

For odd n , i.e. $n = 2a + 1$ for some integer a , the order is given by

$$2p^a \prod_{i=0}^{a-1} (p^{2a} - p^{2i})$$

For even n , i.e. $n = 2a$, the order is given by

$$2(p^a - 1) \prod_{i=0}^{a-1} (p^{2a} - p^{2i}) \quad \text{if } -1 \text{ is a square (mod } p)$$

and

$$2(p^a + (-1)^{a+1}) \prod_{i=1}^{a-1} (p^{2a} - p^{2i}) \quad \text{if } -1 \text{ is a non-square (mod } p)$$

Using the Chinese Remainder Theorem, the order of the group formed by orthogonal matrices over $\mathbb{Z}/m\mathbb{Z}$ where $m=pq$, a square free integer is equal to the product.

(order of orthogonal matrices over $\mathbb{Z}/p\mathbb{Z}$) \times (order of orthogonal matrices over $\mathbb{Z}/q\mathbb{Z}$).

As the factorization of the modulus m is required to calculate the order, this set can be used in the matrix based public key system.

Thus it can be seen that the RSA system can be generalised to matrix rings provided the message space is restricted to avoid nilpotent elements. From a practical implementation point of view, the upper triangular matrices with invertible diagonal elements seems to be the better candidate as the messages can be constructed in an almost arbitrary manner. In the case of non-singular matrices, an additional procedure to find the determinant of the message matrix is required. However, this problem can be overcome by constructing the message matrix as a product of upper triangular and lower triangular matrices as follows:

Let U be an upper triangular matrix and L be a lower triangular matrix with unit diagonal over $\mathbb{Z}/m\mathbb{Z}$. The elements other than the diagonal ones in U and L can be arbitrarily chosen modulo m . As both U and L

are invertible over $\mathbb{Z}/m\mathbb{Z}$, their product $M=LU$ is also invertible over $\mathbb{Z}/m\mathbb{Z}$. Further, the non-commutativity property of matrices ($LU \neq UL$ in general) ensures that the cryptanalyst still needs to factorize m to be able to calculate the decrypting exponent d . This is in contrast to the case of just the upper triangular matrices with unit diagonal mentioned earlier. This is because $M^e = (UL)^e \neq U^e L^e$. Thus although $U^{ed_1} \equiv U \pmod{m}$ and $L^{ed_1} \equiv L \pmod{m}$ where $ed_1 \equiv 1 \pmod{\text{ord}_1}$, $M^{ed_1} \not\equiv M \pmod{m}$ but $M^{ed} \equiv M \pmod{m}$ where $ed \equiv 1 \pmod{N_m \text{ or } E \times P}$. The receiver can recover the matrices L and U uniquely given the matrix M . Furthermore, the above procedure also applies if one of U or L is a triangular matrix with invertible diagonal elements and the other triangular matrix with unit diagonal.

This extended RSA system using matrix messages has been simulated on a Prime Computer [8]. The encryption and the decryption of message matrices have been performed using the Square and Multiply technique [11].

Two points are worth mentioning regarding this extended system. Firstly, it is seen that a non-square free modulus can be used with this system which is not possible with the RSA system over integers. That is, powers of primes can be used to form the modulus m . Secondly, the use of a matrix as a message allows large amounts of data to be processed within one encryption/decryption cycle. Whether this is an advantage depends upon the ease with which matrix manipulation can be carried out in real time.

Ring of Polynomials

Consider the factorization trapdoor system in another ring of special interest, namely the ring of polynomials $R[x]$, which consists of polynomials with coefficients in an arbitrary ring R .

Let $R = \mathbb{Z}/p\mathbb{Z}$ and $f(x)$ be a polynomial in $\mathbb{Z}/p\mathbb{Z}[x]$ of degree s whose factorization is given by

$$f(x) = g_1(x) \dots g_r(x) \pmod{p}$$

where $g_i(x)$, $1 \leq i \leq r$, are distinct irreducible polynomials over $\mathbb{Z}/p\mathbb{Z}$ of degree s_i respectively.

Consider the multiplicative group formed by polynomials over $\mathbb{Z}/p\mathbb{Z}$ of degree less than s and relatively prime to $f(x)$. The order of the group, denoted using the Euler function $\phi_p(f(x))$ is evaluated as follows:

$\phi_p(f(x))$ is equal to the number of invertible elements, that is, units in the residue ring $\mathbb{Z}/p\mathbb{Z}[x]$. This ring is isomorphic to $\mathbb{Z}[x]/(p, f(x))$ and can be regarded as a direct sum of finite fields as

$$\mathbb{Z}[x]/(p, f(x)) \cong \mathbb{Z}[x]/(p, g_1(x)) \oplus \dots \oplus \mathbb{Z}[x]/(p, g_r(x))$$

where $\mathbb{Z}[x]/(p, g_i(x))$ is the finite (Galois) field $GF(p^{s_i})$, s_i being the degree of $g_i(x)$.

Hence

$$\begin{aligned} \text{units of } \left(\frac{\mathbb{Z}[x]}{(p, f(x))} \right) &= \text{units of } \left(\frac{\mathbb{Z}[x]}{(p, g_1(x))} \right) \times \dots \times \text{units of } \left(\frac{\mathbb{Z}[x]}{(p, g_r(x))} \right) \\ &= (p^{s_1} - 1) \dots (p^{s_r} - 1) \end{aligned}$$

Hence

$$\#_p(f(x)) = \prod_{i=1}^r (p^{s_i} - 1) \quad (9)$$

A public key system in $\mathbb{Z}[x]/(p, f(x))$ can therefore be designed as follows [12] : The message space consists of polynomials $\{m(x)\}$ of degree less than s over $\mathbb{Z}/p\mathbb{Z}$. The public encryption key is $(e, p, f(x))$ and the secret decryption key is $(d, p, f(x))$ where the coding exponents e and d are calculated using

$$ed \equiv 1 \pmod{\#_p(f(x))} \quad (10)$$

The encryption procedure raises the message polynomial $m(x)$ to the power e using

$$c(x) \equiv (m(x))^e \pmod{(p, f(x))}$$

The decryption procedure is given by

$$m(x) \equiv (c(x))^d \pmod{(p, f(x))}$$

As the order $\#_p(f(x))$ is dependent on the degrees of the irreducible factors of the modulus polynomial $f(x)$, this scheme provides the trapdoor property.

However, the above scheme is not as secure as the RSA system over rational integers or the matrix based RSA system proposed earlier. This is because the security of this system is dependent on the difficulty of factorizing a composite polynomial into its irreducible factors over a finite field, which in general is not a hard problem in sharp contrast with the factorization problem of a large integer.

Berlekamp [13] proposed an efficient algorithm for factoring polynomials in $\mathbb{Z}/p\mathbb{Z}$. For large primes p , Knuth [11] has suggested some modifications to the Berlekamp's procedure. Once the degrees of the irreducible factors are found, the cryptanalyst can determine the order $\phi_p(f(x))$ and then calculate the secret decoding exponent using (10). Furthermore with this scheme, the same decoding exponent d works for all sets of $g_i(x)$ for $i = 1$ to r , with same degrees s_i .

The security of this system can be increased if it is implemented in the ring $\mathbb{Z}[x]/(m, f(x))$ where m is the product of distinct prime integers $m = \prod_{i=1}^t p_i$ and $f(x)$ is a square free composite polynomial as before.

The message space then consists of polynomials $\{m(x)\}$ of degrees less than s with coefficients in $\mathbb{Z}/m\mathbb{Z}$. Using the Chinese Remainder Theorem, the ring $\mathbb{Z}[x]/(m, f(x))$ is isomorphic to the direct of sum of rings given below

$$\frac{\mathbb{Z}[x]}{(m, f(x))} \cong \frac{\mathbb{Z}[x]}{(p_1, f(x))} \oplus \dots \oplus \frac{\mathbb{Z}[x]}{(p_t, f(x))}$$

The order of the multiplicative group formed by polynomials of degrees less than s and relatively prime to $f(x)$ is equal to the number of units in $\mathbb{Z}[x]/(m, f(x))$ and is given by $\phi_m(f(x))$

$$\begin{aligned} \phi_m(f(x)) &= \text{units of } \left(\frac{\mathbb{Z}[x]}{(m, f(x))} \right) \\ &= \text{units of } \left(\frac{\mathbb{Z}[x]}{(p_1, f(x))} \right) \times \dots \times \text{units of } \left(\frac{\mathbb{Z}[x]}{(p_t, f(x))} \right) \end{aligned}$$

Hence

$$\phi_m(f(x)) = \prod_{i=1}^t \phi_{p_i}(f_i(x))$$

where

$$f_i(x) \equiv f(x) \pmod{p_i} \quad 1 \leq i \leq t.$$

Let the factorization of $f_i(x)$ be

$$f_i(x) = \prod_{j=1}^{n_i} g_{ij}(x) \pmod{p_i} \quad (11)$$

where the degree of irreducible polynomial $g_{ij}(x)$ over $\mathbb{Z}/p_i\mathbb{Z}$ is s_{ij} . The upper limit in the product term in (11) goes up to n_i as it is a function of t to which prime p_i the polynomial $f(x)$ is being factored. This is because in general $f(x) \pmod{p_i}$ will have some n_i distinct irreducible factors as i varies.

But using (9)

$$\phi_{p_i}(f_i(x)) = \prod_{j=1}^{n_i} (p_i^{s_{ij}} - 1)$$

Hence,

$$\phi_m(f(x)) = \prod_{i=1}^t \prod_{j=1}^{n_i} (p_i^{s_{ij}} - 1)$$

The order now depends not only on the degrees of the irreducible factors but also on the prime divisors of modulus m . Thus the cryptanalyst needs to factorize both m and $f(x)$ and this gives rise to a system which is at least as strong as the corresponding RSA system over the integers. Furthermore, from cryptography point of view, it is required that both $f(x)$ and m must be square free to avoid nilpotent elements and enable proper decryption. In this respect, it differs from the matrix RSA system described earlier. This system has also been simulated on the Prime Computer[8].

Ring of Algebraic Integers

We now consider the design of public key systems in some algebraic number fields based on factorization trapdoor.

A number θ is said to be an algebraic number [14] if it satisfies a polynomial equation

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0$$

where the coefficients a_i are rational numbers, Q .

If the equation has rational integer coefficients and it is monic, then θ is said to be an algebraic integer. If θ is algebraic over Q , then the field $K = Q(\theta)$ is defined to be the smallest extension field containing both Q and θ . That is, it consists of numbers of the form

$$a_0 + a_1 \theta + a_2 \theta^2 + \dots + a_n \theta^n$$

where a_i are rational numbers.

The subset of the field K consisting of algebraic integers forms a ring D , called the ring of algebraic integers in K . In general, D is not a unique factorization domain. Factorization of elements in D is unique if and only if every irreducible in D is also a prime, that is, if and only if D is a principal ideal domain (PID). The rings where unique factorization of integers fails correspond to non-principal ideal domains. In such domains, there are irreducibles which are not primes and they generate principal ideals which are not prime ideals but factorize into non-principal ideals [15]. We only consider the design of the factorization trapdoor systems in PIDs. Unless otherwise stated, from now on D is assumed to be a PID.

Let m be a square free integer in some ring D and its factorization be

$$m = \prod_{i=1}^r \pi_i$$

where π_i are irreducibles or primes in D .

Then, using the Chinese Remainder Theorem, the residue class ring is isomorphic to the direct sum of finite fields as

$$D/\langle m \rangle \cong D/\langle \pi_1 \rangle \oplus \dots \oplus D/\langle \pi_r \rangle$$

where $\langle \pi \rangle$ denotes the principal ideal generated by π . The order of the group formed by invertible residue classes modulo the ideal $\langle m \rangle$ is given by $\phi \langle m \rangle$, which is similar to the Euler function ϕ for rational integers.

$$\phi \langle m \rangle = \phi \langle \pi_1 \rangle \cdot \dots \cdot \phi \langle \pi_r \rangle \quad (12)$$

For a prime π_i ,

$$\phi \langle \pi_i \rangle = N \langle \pi_i \rangle - 1 \quad (13)$$

where $N \langle \pi_i \rangle$ denotes the norm of the ideal $\langle \pi_i \rangle$, the number of residue classes modulo the ideal $\langle \pi_i \rangle$.

A public key system is therefore possible as the order depends on the prime factors of m . Such a scheme is illustrated by considering a simple quadratic field $K = Q(i)$. The ring of integers $D = Z[i]$ consists of elements of the form $\{a + bi \mid a, b \in Z\}$ and is commonly known as the ring of Gaussian integers. Now m is a composite integer in $Z[i]$ and its factors π_i , $1 \leq i \leq r$, are primes in $Z[i]$. To be able to calculate $\phi \langle m \rangle$, it is necessary to find $N \langle \pi_i \rangle$, $1 \leq i \leq r$, (see (12) and (13)).

The norm $N\langle\pi_i\rangle$ is a rational integer and is equal to

$$N\langle\pi_i\rangle = \pi_i \overline{\pi_i}$$

where $\overline{\pi_i}$ is the complex conjugate of π_i .

Let the prime decomposition of $N\langle\pi_i\rangle$ in \mathbb{Z} be

$$N\langle\pi_i\rangle = p_1 \cdots p_t$$

where p_i are distinct primes in \mathbb{Z} .

Then, as $\pi_i \mid N\langle\pi_i\rangle$, $\pi_i \mid p_1 \cdots p_t$. That is, π_i divides one of the primes p_j . It cannot divide two primes p_j and p_k . If so, then it is possible to find two integers a and b using Euclid's algorithm such that $ap_j + bp_k = 1$. As $\pi_i \mid p_j$ and $\pi_i \mid p_k$, $\pi_i \mid 1$. So π_i is a unit, not a prime which is contrary to the assumption. Thus every Gaussian prime π_i divides only one rational prime p_i . Hence $N\langle\pi_i\rangle$ divides Np_i . But $Np_i = p_i^2$. Therefore, $N\langle\pi_i\rangle = p_i$ or p_i^2 . It can be shown that [14], if $p_i \equiv 1 \pmod{4}$ then $N\langle\pi_i\rangle = p_i$ whereas if $p_i \equiv 3 \pmod{4}$ then $N\langle\pi_i\rangle = p_i^2$. Thus the order $\phi\langle m \rangle$ is given by

$$\phi\langle m \rangle = \prod_{i=1}^r N\langle\pi_i\rangle - 1$$

where

$$N\langle\pi_i\rangle = \begin{cases} p_i & p_i \equiv 1 \pmod{4} \\ p_i^2 & p_i \equiv 3 \pmod{4} \end{cases}$$

The encryption and decryption coding exponents e and d can be calculated using

$$ed \equiv 1 \pmod{\phi(N)} \quad (14)$$

The messages are represented using the residue classes modulo the ideal $\langle m \rangle$ and there are $N\langle m \rangle$ such residue classes.

Case 1

First consider the case where the primes π_i which form m divide rational primes p_i of the form $p_i \equiv 1 \pmod{4}$. Then the norm is a square free rational integer given by

$$\begin{aligned} N\langle m \rangle &= \prod_{i=1}^r N\langle \pi_i \rangle \\ &= \prod_{i=1}^r p_i \end{aligned}$$

The residue class ring $Z[i]/\langle m \rangle$ is isomorphic to the direct sum of finite fields $Z[i]/\langle \pi_i \rangle$, $1 \leq i \leq r$. The field $Z[i]/\langle \pi_i \rangle$ contains p_i elements. Therefore, one standard method of representing the messages mod $\langle m \rangle$ would be to use the integers in the ring $Z/N\langle m \rangle Z$, that is, 0 to $N\langle m \rangle - 1$. This is similar to the message space of the RSA system over rational integers. The encryption and decryption processes are carried out using $C \equiv M^e \pmod{N\langle m \rangle}$ and $M \equiv C^d \pmod{N\langle m \rangle}$ and $(e, N\langle m \rangle)$ is the public key.

Now consider the situation where the message space still consists of the integers in $Z/N\langle m \rangle Z$ but the encryption and decryption procedures are calculated modulo m , $m \in Z[i]$. Let $m = a+bi$ and the message be M in $Z/N\langle m \rangle Z$. Then encryption gives, say,

$$C \equiv M^e \equiv g + hi \pmod{(a+bi)}$$

Decryption produces

$$(g+hi)^d \equiv k + li \pmod{(a+bi)}$$

That is, the recovered message M is equal to $k + li$

$$M \equiv k + li \pmod{(a+bi)} \quad (15)$$

Conjugating both sides of (15)

$$M \equiv k - li \pmod{(a-bi)}$$

Using Chinese Remainder Theorem, the original M can be obtained as

$$M \equiv \alpha_1(k-li) + \alpha_2(k+li) \pmod{(a+bi)(a-bi)}$$

where $\alpha_1 + \alpha_2 = 1$ $\alpha_1, \alpha_2 \in \mathbb{Z}[i]$.

Case 2

If the primes \prod_i which form m divide rational primes p_i of the form $p_i \equiv 3 \pmod{4}$, then the $N\langle m \rangle$ is a non-square free rational integer given by

$$N\langle m \rangle = \prod_{i=1}^r p_i^2$$

In this case, although $\mathbb{Z}[i]/\langle \prod_i \rangle$ is a finite field of p_i^2 elements, one cannot represent the residue classes modulo $\langle \prod_i \rangle$ using the integers $\mathbb{Z}/p_i^2\mathbb{Z}$ as the latter does not form a finite field. On the other hand, one can represent the messages in the form $M = x+iy$ where $0 < x, y < \left\lfloor \sqrt{N\langle m \rangle} \right\rfloor - 1$, thus giving rise to $N\langle m \rangle$ distinct residue classes modulo $\langle m \rangle$.

Encryption is performed by raising the message M to the power e and reducing the coefficients modulo $\left\lfloor \sqrt{N\langle m \rangle} \right\rfloor$. That is, if $M = x + iy$, then

$$\begin{aligned} C &\equiv M^e \equiv (x+iy)^e \pmod{\langle m \rangle} \\ &\equiv g \left(\pmod{\left\lfloor \sqrt{N\langle m \rangle} \right\rfloor} \right) + h \left(\pmod{\left\lfloor \sqrt{N\langle m \rangle} \right\rfloor} \right) i \end{aligned}$$

A similar procedure is carried out in decryption.

Case 3

If m factorizes into primes π_i some of which divide rational primes $p \equiv 1 \pmod{4}$ and others divide rational primes $p \equiv 3 \pmod{4}$, then it can be shown [8] that the cryptanalyst can easily partly factorize m and hence reduce the difficulty of breaking the system. Therefore from cryptography point of view, this case should not be used.

The security of the public key system in $Z[i]$ again depends on the difficulty of factorizing a large rational integer into its primes; in Case 1, the rational integer $N\langle m \rangle = p_1 \dots p_r$ needs to be factored whereas in Case 2, the rational integer $|\sqrt{N\langle m \rangle}| = p_1 \dots p_r$ needs to be factored. In both cases, once the primes p_1 to p_r are found, then the order $\phi\langle m \rangle$ can be easily determined using
$$\phi\langle m \rangle = \prod_{i=1}^r N\langle \pi_i \rangle - 1 \quad \text{where } N\langle \pi_i \rangle = p_i \text{ or } p_i^2.$$
 Then, the secret coding exponent d can be calculated using (14). Note that the cryptanalyst does not need to know the Gaussian primes π_1 to π_r but only needs to know their respective norms. In other words, the cryptanalyst will be working over Z and not over $Z[i]$.

The design of factorization trapdoor system as described above can be extended to other quadratic fields which are principal ideal domains.

Discussion

A generalization of the RSA cryptosystem in the ring of matrices over Z/mZ where m is a composite integer is proposed. It is shown that

factorization of the modulus m is needed to compute the order of the group formed by non-singular matrix messages, upper triangular matrix messages with non-unity invertible diagonal elements and orthogonal matrix messages thus offering a similar level of security as the RSA system.

An extension of the RSA system to polynomial rings has been considered. The difficulty of factorization of a polynomial into its irreducible factors over a finite field does not in itself provide a secure public key cryptosystem. However if the difficulty of factorizing a polynomial is compounded with the difficulty of factorizing an integer then a secure RSA type cryptosystem in the ring of polynomials is seen to be possible.

The design of public key system in some quadratic algebraic number fields using factorization trapdoor concept has been presented. The security of such systems is found to be dependent on the difficulty of factoring the norm of the modulus.

The investigation of such extensions of RSA cryptosystem indicates that rings other than the ring of rational integers can be used to construct public key systems based on factorization trapdoor property. From a practical point of view, however it seems that the complexity of such systems may favour the implementation of the factorization trapdoor in the ring of rational integers.

References

1. Diffie, W. and Hellman, M.E., 'New Directions in Cryptography', IEEE Trans. on Inf. Theory, Vol. IT-22, 1976, pp 644-654.
2. Rivest, R.L., Shamir, A. and Adleman, L., 'A method for obtaining Digital Signatures and Public Key Cryptosystems', Comm. ACM, Vol. 21, No. 2, 1978, pp 120-126.
3. Diffie, W. and Hellman, M.E., 'Privacy and Authentication : An Introduction to Cryptography', Proc. IEEE, Vol. 67, NO. 3, 1979.
4. Davies, D., 'Limits to Computations', NPL note, London.
5. Van der Waerden, B.L., Modern Algebra : Vol. 1 and 2, 1949.
6. Dickson, L.E., Linear Groups with an Exposition of the Galois Field Theory, Dover Pub., 1958.
7. Albert, A.A., Fundamental Concepts of Higher Algebra, The Univ. of Chicago Press, 1956.
8. Varadharajan, V., Some Cryptographic Techniques for Secure Data Communication, Ph.D. Thesis, CNAA, 1984.
9. Varadharajan, V. and Odoni, R., 'Extension of RSA cryptosystem to Matrix Rings', Cryptologia, Accepted for Publication Aug. 1984.
10. MacWilliams, J., 'Orthogonal matrices over Finite Fields', American Mathematical Monthly, Feb. 1969.
11. Knuth, D.E., The Art of Computer Programming, Vol. 2 : Seminumerical Algorithms, Second Edition, Addison-Wesley, 1981.
12. Kravitz, D.W. and Reed, I.S., 'Extension of RSA Cryptosystem : A Galois Approach', IEE Electronic Letters, Vol. 18, No. 6, 1982, pp 255-256.
13. Berlekamp, E.R., 'Factoring Polynomials over large Finite Fields', Maths. of Computation, Vol. 24, No. 111, 1970, pp 713-735.
14. Pollard, H., The Theory of Algebraic Numbers, The Carus Math. Monographs, No. 9, Pub. by Math. Assoc. of America, John Wiley, 1950.
15. Rosen, M. and Ireland, K., A Classical Introduction to Modern Number Theory, Springer-Verlag, 1980.

Acknowledgements

The author would like to acknowledge the help of Prof. R. Odoni, Dept. of Maths., Exeter University, for valuable discussions on the subject.

ON COMPUTING LOGARITHMS OVER FINITE FIELDS

TaberElGarnal

Hewlett-Packard Labs
3172 Porter Dr., bldg 29U
Palo Alto CA 94304

ABSTRACT

The problem of computing logarithms over finite fields has proved to be of interest in different fields [4]. Subexponential time algorithms for computing logarithms over the special cases $GF(p)$, $GF(p^2)$ and $GF(p^m)$ for a fixed p and $m \rightarrow \infty$ have been obtained. In this paper, we present some results for obtaining a subexponential time algorithms for the remaining cases $GF(p^m)$ for $p \rightarrow \infty$ and fixed $m \neq 1, 2$. The algorithm depends on mapping the field $GF(p^m)$ into a suitable cyclotomic extension of the integers (or rationals). Once an isomorphism between $GF(p^m)$ and a subset of the cyclotomic field $\mathbb{Q}(\omega_q)$ is obtained, the algorithm becomes similar to the previous algorithms for $m = 1, 2$.

A rigorous proof for subexponential time is not yet available, but using some heuristic arguments we can show how it could be proved. If a proof would be obtained, it would use results on the distribution of certain classes of integers and results on the distribution of some ideal classes in cyclotomic fields.

1. INTRODUCTION

This paper gives some ideas for extending the Merkle - Adleman algorithm for computing discrete logarithms over $GF(p)$ [1,7,9] to higher order fields. Section 2 finds appropriate integral domains for extending the algorithm. The reader is referred to [8,11] for discussion on number fields and using integral domains to extend the algorithm. Section 3 gives some ideas regarding the running time of the algorithm.

2. FINDING THE ISOMORPHISM:

From the discussion in [8], it seems natural to use higher number fields to extend the algorithm to higher order finite fields. Unfortunately, higher algebraic number fields do not have all the properties of quadratic fields that were used in proving a subexponential running time in [8]. For example, the norm function is not as easy to find, and hence the proofs for the fraction of smooth elements are more difficult. So the discussion in this paper is restricted to using a certain class of algebraic number fields; namely, the cyclotomic fields. For a discussion of the properties of cyclotomic fields, the reader is referred to [11]. Cyclotomic fields are used because they possess some of the properties of quadratic fields that were needed in developing the algorithm for the case $GF(p^2)$. For example, the splitting of primes in cyclotomic extensions is easy to determine, which is not the case for general fields.

For simplicity, only "prime" cyclotomic fields will be used, i.e. the fields $\mathbb{Q}(\omega_q)$ where ω_q is a primitive q th root of unity, and q is a prime in \mathbb{Z} . The q th cyclotomic polynomial has the form

$$\Phi_q(D) = D^{q-1} + D^{q-2} + \dots + D + 1.$$

Note that the general cyclotomic polynomial does not necessarily have this nice form. Hence, the q th cyclotomic field has degree $q-1 = \phi(q)$. Some results on cyclotomic fields are needed to find the appropriate cyclotomic fields. The reader is referred to [11] for proofs.

Recall, from [11], the results on the splitting of primes in cyclotomic extensions (known as Kummer's theorem). For each prime $p \in \mathbb{Z}$

$$(p) = \prod_{i=1}^g (p, h_i(\omega_q)),$$

where

$$\Phi_p(D) = \prod_{i=1}^g h_i(D) \pmod{q}.$$

The polynomials $h_i(D)$ all have degree f , where $fg = q - 1$, and f is equal to the order of $p \pmod{q}$ (or the order of p in the multiplicative group in $GF(q)$, which is usually denoted by $(\mathbb{Z}/q)^*$). Hence the splitting of the ideal (p) in $\mathbb{Z}(\omega_q)$ depends on the factorization of the p th cyclotomic polynomial \pmod{q} which is easy to find (see [11]).

If $R_i = (p, h_i(\omega_q))$, then $N(R_i) = p^f$, where $N(R_i)$ is the norm of the ideal R_i .

The next lemma relates cyclotomic polynomials to the orders of elements in $(\mathbb{Z}/q)^*$.

Lemma 1

Let q be a prime $\leq n$, and let $a \in \mathbb{Z}$. Then $q \mid \Phi_n(a)$ if and only if the order of a in $(\mathbb{Z}/q)^*$ is n .

Proof

First, if the order of $a \pmod{q}$ is equal to n , then $a^n - 1 = 0 \pmod{q}$ and n is the smallest such exponent. Hence, q divides one of the factors of the polynomial $D^n - 1$ evaluated at $D = a$. It is known that $D^n - 1 = \prod_{d|n} \Phi_d(D)$ (see [11]). Hence, q divides $\Phi_n(a)$ since, if it divides another factor of $a^n - 1$, then its order is less than n . Conversely, if q divides $\Phi_n(a)$ then $a^n - 1 = 0 \pmod{q}$ since q divides one of the factors of the polynomial $D^n - 1$ evaluated at $D = a$, and n is the smallest such exponent (otherwise q would divide $\Phi_d(a)$ for some $d < n$ in which case the polynomial $D^n - 1$ has multiple roots which is never the case [11]). This proves Lemma 1.

This lemma provides an easy check for the order of $p \pmod{q}$. That is, if the order of $p \pmod{q}$ is equal to f , then q has to divide $\Phi_f(p)$.

Going back to the isomorphism, a cyclotomic field $\mathbb{Q}(\omega_q)$ is used to generate a finite field $GF(p^m)$, for p and m known (and m small). A field that is isomorphic to $GF(p^m)$ needs to be found from the ring of integers in a cyclotomic field similar to the isomorphisms that were found for the cases $m = 1, 2$.

One observation is that if the "residue classes" $\mathbb{Z}(\omega_q)/R_i$ for some prime ideal R_i of norm p^m are constructed, then these residue classes form a finite field isomorphic to $GF(p^m)$. Let

$$R_i = (p, h_i(\omega_q)),$$

and

$$\Phi_q(D) = \prod_{i=1}^g h_i(D) \pmod{p},$$

where each $h_i(D)$ is irreducible \pmod{p} . Then $h_i(D)$ is a candidate for generating $GF(p^m)$.

Finding the appropriate field $\mathbb{Q}(\omega_q)$

The discrete logarithm problem is the following; given α, y and p^m , find x such that

$$\alpha^x = y \text{ in } GF(p^m)$$

for some given irreducible polynomial $K(D)$ with degree m . First, as noted in [2,3,10], the choice of the irreducible $K(D)$ does not affect the running time of the algorithm since all representations of $GF(p^m)$ are isomorphic and only polynomial time is needed to find the corresponding logarithms in one representation if the logarithms are known in another representation.

From the above discussion, a prime ideal R that has norm equal to p^m needs to be obtained. That is equivalent to finding a prime q such that p has order $m \pmod{q}$.

Equivalently, to construct an appropriate field $\mathbb{Q}(\omega_q)$, a prime factor q of $\Phi_m(p)$ should be computed (see Lemma 1). This proves the existence of such q , which might be quite large (for example $O(p)$ or higher). In this case the obtained field cannot be used for our algorithm since just representing an integer takes $O(p)$ operations.

Fortunately, as p grows larger the probability that $\Phi_m(p)$ has at least one small factor is high if the number $\Phi_m(p)$ is assumed to be random, but for some given p and m no small divisor q may exist. The reason is that $p^m - 1$ should be chosen to have at least one large prime

factor, and hence $\Phi_m(p)$ (which is a factor of $P^m - 1$) is likely to be a prime, or not to have any large prime factor.

For the cases where $\Phi_m(p)$ does not have any small factors, $GF(p^m)$ could be embedded in $GF(p^{im})$ for some small $i \in \mathbb{Z}$. $\log y$ can be found as if y and α were elements in $GF(p^{im})$ and the results are transferred back to $GF(p^m)$ which is isomorphic to the subfield of order p^m in $GF(p^{im})$.

So in this case, a small divisor of $\Phi_{im}(p)$ for some $i \in \mathbb{Z}$ is needed. That increases the chance of finding an appropriate q , since the probability that one of the numbers $\Phi_{im}(p)$, $i = 1, 2, \dots, l$ for some l , has at least one small prime factor grows with l .

Note that $\Phi_m(p)$ need not be factored completely because only a small divisor ($O(\log p)$ for example) is needed. Even if $\Phi(p)$ is factored completely, the asymptotic running time of algorithm will not increase since $\Phi_m(p) = O(p^{e(m)})$ and factoring such a number also takes subexponential time in $\varphi(m) \log p$.

3. THE RUNNING TIME

This section sketches some ideas about the running time of the algorithm as described above.

A. The image of $GF(p^m)$ is $\mathbb{Z}(\omega_q)/\mathbf{A}$ which consists of the elements

$$\left\{ \sum_{j=0}^{m-1} a_j \omega^j, a_j \in \mathbb{Z} \text{ for all } j \right\}.$$

and the norm of the ideal \mathbf{A} is p^m .

B. All the elements in $\mathbb{Z}(\omega_q)/\mathbf{A}$ have norm less than

$$M = m^2 (p-1)^m.$$

This is a loose bound, since it is obtained by adding m^2 terms. (When computing the norm of any element in $\mathbb{Z}(\omega_q)$, m^2 are obtained, each has the value $(p-1)^m$ which is the maximum value of each term, since each α_i is less than p .)

C. The number of ideals in $\mathbb{Z}(\omega_q)$ with norm up to M is linear in M ($= kM$) for some constant k (see [8]).

The number of prime ideals in $\mathbb{Z}(\omega_q)$ with norm up to M is therefore equal to $O\left(\frac{M}{\log M}\right)$.

D. The number of principal prime ideals up to norm M is equal to the total number of prime ideals with norm up to $M^{1/h}$, where h is the class number of $\mathbb{Z}(\omega_q)$, because any ideal $A \in \mathbb{Z}(\omega_q)$ raised to the h th power is principal.

E. The number of smooth principal ideals in $\mathbb{Z}(\omega_q)$ with norm up to M (smooth is defined with respect to some value for the maximum norm of small prime principal ideals N) can be computed in a way similar to the computation in [6] for the case of $GF(p^2)$.

F. Assume that the smooth elements are uniformly distributed among the different subsets of elements with small norm. Then, the ratio of smooth elements in $\mathbb{Z}(\omega_q)/R$ is of the same form as for the cases $GF(p)$ and $GF(p^2)$, and a subexponential running time could be obtained.

REFERENCES

- [1] L. Adleman, "A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography" to be published.
- [2] I. Blake, R. Fuji-Hara, R. Mullin, and S. Vanstone, "Computing Logarithms in Finite Fields of Characteristic Two", *to be published*.
- [3] D. Coppersmith, "Fast Evaluation of Logarithms in Fields of Characteristic Two", *to appear in IEEE Transactions on Information Theory*, July 1984.
- [4] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, vol. IT-22 pp.644-654 Nov. 1976.
- [5] W. Diffie and M. Hellman, "Privacy and Authentication: An Introduction to Cryptography", *Proceedings of the IEEE*, vol 67, No 3, March 1979.
- [6] T. ElGamal, "A Subexponential-Time Algorithm for Computing Discrete Logarithms over $GF(p^2)$ ", *submitted to IEEE Transactions on Information Theory*.
- [7] M. Hellman and J. Reyneri, "Fast Computation of Discrete Logarithms in $GF(p^m)$ ", "

Presented at Crypto 82 Conference Santa Barbara, CA August 1982.

- [8] D. Marcus, *Number Fields*, Springer-Verlag.
- [9] R. Merkle, *Secrecy, Authentication, and Public Key Systems*, Ph.D. Dissertation, Electrical Engineering Department, Stanford University June 1979.
- [10] A. Odlyzko, "Discrete Logarithms in Finite Fields and Their Cryptographic Significance",
to be published. Journal of Number Theory vol. 15 no. 2, October 1982.
- [11] L. C. Washington, *Introduction to Cyclotomic Fields*, Graduate texts in mathematics 83.
Springer - Verlag 1982.

ON USING RSA WITH LOW EXPONENT IN A PUBLIC KEY NETWORK

by Johan Hastad*
MIT

Abstract: We consider the problem of solving systems of equations $P_i(x) \equiv 0 \pmod{n_i}$ $i = 1 \dots k$ where P_i are polynomials of degree d and the n_i are distinct relatively prime numbers and $x < \min n_i$. We prove that if $k > \frac{d(d+1)}{2}$ we can recover x in polynomial time provided $n_i \gg 2^k$. This shows that RSA with low exponent is not a good alternative to use as a public key cryptosystem in a large network. It also shows that a protocol by Broder and Dolev [4] is insecure if RSA with low exponent is used.

1. Introduction

Let us start with some cryptographic motivation. The famous RSA function [8] is defined as $f(x) \equiv x^d \pmod{n}$. Here n is usually taken of the form $n = pq$ where p and q are two large primes and d is an integer relatively prime to $(p-1)(q-1)$. Using these parameters the function is 1-1 when restricted to $1 \leq x \leq n$, $(x, n) = 1$. Furthermore the function is widely believed to be a trapdoor function i.e. given n and d it is easy to compute $f(x)$ and given $f(x)$ it is also easy to recover x provided you have some secret information but otherwise it is infeasible. In this case the secret information is the factorization of n .

The RSA function can be used to construct a deterministic Public Key Cryptosystem(PKC) in the following way:

Each user B in a communication network chooses two large primes p and q and multiplies them together and publishes the result n_B together with a number d_B which is relatively prime to $(p-1)(q-1)$. He keeps the factorization as his private secret information. If any user A in the system wants to send a secret message m to another user B she retrieves B 's published information computes $y \equiv m^{d_B} \pmod{n_B}$ and sends y to B . B now obtains the original message using his secret information while somebody else presumably faces an intractable computational task.

However PKC are different and more complex objects than trapdoor functions. For example the use of RSA in a PKC may present obstacles that did not occur when we considered it as a trapdoor function. Several people (at least Blum, Lieberherr and Williams) have observed the following attack. Assume that 3 is chosen as the exponent and that A wants to send the same message m to users U_1, U_2 and U_3 . She will compute and send $y_i \equiv m^3 \pmod{n_i}$ $i = 1, 2, 3$. But using the fact that n_1, n_2 and n_3 are relatively prime a listener who know the values of y_1, y_2 and y_3 can combine the messages by chinese remaindering to get $m^3 \pmod{n_1 n_2 n_3}$ and since $m^3 < n_1 n_2 n_3$ he can recover m . In general if the exponent is d the number of messages needed is d .

* Supported by an IBM fellowship, partially supported by NSF grant DCR-8509905

A natural question is therefore: Is there a better way to send the same message to many people using this PKC?

A common heuristic tells us to use a "time stamp". Instead of sending the same message m to everybody one attaches the time and thus sends the encryption of $2^{[t]}m + t$ where $2^{[t]}m$ is the shifted message and t is the time (which will be different for the different receivers). The previous attack fails and we are led to the following computational problem (for $d = 3$).

Given $(a_i m + b_i)^3 \pmod{n_i}$ where all the a_i and b_i are known is it possible to recover m in polynomial time?

We will see in section 3 that the answer is YES if the number of similar messages is at least 7. In fact we will prove that given a set of equations

$$P_i(x) \equiv 0 \pmod{n_i} \quad i = 1, \dots, k$$

where we have k polynomial equations of degree $\leq d$ it is possible to recover the solution in time polynomial in both k and $\log n_i$ if $k > d(d+1)/2$ provided $n_i \gg 2^d$. Therefore we conclude that if RSA is to be used as a PKC we should use a large exponent or even better use a probabilistic encryption scheme [3],[6] based on RSA. By [1],[3] this can be done with as much efficiency as in the deterministic case.

2. The insecurity of a protocol by Broder and Dolev.

Broder and Dolev proposed a protocol for flipping a coin in a distributed system [4]. Some of their essential ingredients were Shamir's method of sharing a secret and the use of a deterministic PKC. They proposed to use the RSA. In [2] it is shown that what they really need from the security of the cryptosystem is:

Given the encryption of $a_i x + b_i$ with different keys it should be infeasible to decide the parity of x with a better probability than flipping a coin. The analysis in the next section shows that given this information we can, not only find the parity of x , but the exact value of x if the PKC is RSA with a small exponent. In the case of a large exponent the protocol is not known to be insecure but on the other hand there is no proof of correctness. A provably secure protocol has been designed by Awerbuch et.al. [2].

3. Main Theorem

Let us start by fixing some notation. Let $N = \prod_{i=1}^k n_i$ and $n = \min n_i$. Now we can state the problem formally:

Problem: Given a set of k equations $\sum_{j=0}^d a_{ij} x^j \equiv 0 \pmod{n_i}$, $i = 1, \dots, k$. Suppose that the system have a solution $x < n$. Can we find such a solution efficiently?

Before we give the theorem let us give the basic ideas. Define $u_j < N$ to be the chinese remaindering coefficients i.e. $u_j \equiv \delta_{ij} \pmod{n_i}$ ($\delta_{ij} = 1$ if $i = j$ and 0 otherwise). We can combine the equations to a single equation using the chinese remainder theorem.

$$0 \equiv \sum_{j=0}^d x^j \sum_{i=1}^k u_i a_{ij} \equiv \sum_{j=0}^d x^j c_j \pmod{N}$$

One of the important parts of the entire paper is the following simple lemma.

Lemma 1: If $|c_j| < \frac{N}{(d+1)n^j}$ we can find x in time polynomial in d, k and $\log n_i$.

Proof: If $|c_j| < \frac{N}{(d+1)n^j}$ then

$$\left| \sum_{j=0}^d c_j x^j \right| \leq \sum_{j=0}^d |c_j| n^j < N$$

Thus the condition $\sum_{j=0}^d c_j x^j \equiv 0 \pmod{N}$ implies $\sum_{j=0}^d c_j x^j = 0$. In other words x solves the equation over the integers and to prove the lemma we just need the fact that we can solve polynomial equations over the integers in polynomial time. This follows from [7] but there are more efficient algorithms.

The condition of lemma 1 is quite unlikely to be fulfilled when we start with a general set of equations. In spite of this lemma 1 will be one of our main tools for proving:

Theorem: Given a set of equations $\sum_{j=0}^d a_{ij} x^j \equiv 0 \pmod{n_i}$, $i = 1, 2, \dots, k$ where $x < n$ and $\gcd(\langle a_{ij} \rangle_{j=0}^d, n_i) = 1$ for all i . Then it is possible to recover x in time polynomial in d, k and $\log n_i$ if

$$N > n^{\frac{d(d+1)}{2}} (k+d+1)^{\frac{k+d+1}{2}} 2^{\frac{(k+d+1)^2}{2}} (d+1)^{(d+1)}$$

As before $N = \prod_{i=1}^k n_i$, $n = \min n_i$, d is the degree of the equations and k is the number of equations.

Proof: The idea is to use lemma 1. However as we remarked it is quite unlikely that it will apply to our equations directly. To get more possibilities we will multiply the i -th equation by a constant s_i before we combine them using chinese remaindering. If we have chosen the s_i carefully enough the resulting equation will have the desired small coefficients. We get

$$\sum_{j=0}^d x^j \sum_{i=1}^k a_{ij} s_i u_i \equiv 0 \pmod{N}$$

Let c_j denote the coefficient of x^j in this equation. To apply lemma 1 we want $|c_j| n^j < \frac{N}{d+1}$. The main tool for achieving this will be the use of lattices. We first start by recalling some background from the geometry of numbers.

3.1 Background from geometry of numbers.

A lattice L is defined to be the set of points

$$L = \{ y \mid y = \sum_{i=1}^n a_i \vec{b}_i, a_i \in \mathbb{Z} \}$$

where \vec{b}_i are linearly independent vectors in \mathbb{R}^n . The set \vec{b}_i is called a basis for the lattice and n is the dimension. The determinant of a lattice is defined to be the absolute value of the determinant of the matrix with rows \vec{b}_i . It is not hard to see that the determinant is independent of the choice of basis. The length of the shortest nonzero vector in the lattice is denoted by λ_1 . Let us recall the following wellknown facts:

Theorem: (Minkowski) $\lambda_1 \leq \gamma_n^{\frac{1}{n}} (\det(L))^{\frac{1}{n}}$ where γ_n is Hermite's constant.

γ_n is not known explicitly but we have an upper bound $\gamma_n \leq n$ [5].

Theorem: We can find a vector \vec{b} in polynomial time which satisfies $\|\vec{b}\|/\lambda_1 \leq 2^{\frac{1}{2}}$.

This is bound you get from the famous algorithm in the paper by Lenstra, Lenstra and Lovasz [7]. By a result by Schnorr it is possible to replace the constant 2 by any number greater than 1 [9] but this is not important to us. Armed with this information we return to the original problem.

3.2 Continuation of Proof.

Define the following lattice L of dimension $k + d + 1$ by its base vectors:

$$\vec{b}_1 = (a_{10}u_1, na_{11}u_1, n^2a_{12}u_1, \dots, n^da_{1d}u_1, \frac{N}{n_1(d+1)}, 0, \dots, 0)$$

$$\vec{b}_2 = (a_{20}u_2, na_{21}u_2, n^2a_{22}u_2, \dots, n^da_{2d}u_2, 0, \frac{N}{n_2(d+1)}, \dots, 0)$$

$$\vec{b}_k = (a_{k0}u_k, na_{k1}u_k, n^2a_{k2}u_k, \dots, n^da_{kd}u_k, 0, 0, \dots, \frac{N}{n_k(d+1)})$$

$$\vec{b}_{k+1} = (N, 0, 0, \dots, 0, 0, 0, \dots, 0)$$

$$\vec{b}_{k+2} = (0, nN, 0, \dots, 0, 0, 0, \dots, 0)$$

$$\vec{b}_{k+d+1} = (0, 0, 0, \dots, n^dN, 0, 0, \dots, 0)$$

Observe that

$$\begin{aligned} \sum_{i=1}^k s_i \vec{b}_i &= \left(\sum_{i=1}^k s_i a_{i0} u_i + s_{k+1} N, n \sum_{i=1}^k s_i a_{i1} u_i + s_{k+2} N, \dots, \frac{Ns_1}{n_1(d+1)}, \dots, \frac{Ns_k}{n_k(d+1)} \right) \equiv \\ &\equiv (c_0, nc_1, n^2c_2, \dots, \frac{Ns_1}{n_1(d+1)}, \dots, \frac{Ns_k}{n_k(d+1)}) \pmod{N} \end{aligned}$$

Observe that for $1 \leq i \leq k+1$ the i 'th coefficient is divisible by n^i . We multiply the different coefficients by the corresponding powers of n since we want $|c_j|n^j < \frac{N}{d+1}$. The last k coordinates are there to make the multipliers s_i small in a short vector in the lattice and the last $d+1$ vectors reflect the fact that we have a modular equation.

The only term in the expansion of the determinant is the diagonal term and we get

$$\det(L) = n^{\frac{d(d+1)}{2}} N^{d+k+1} (d+1)^{-k} \prod_{i=1}^k n_i^{-1} = n^{\frac{d(d+1)}{2}} N^{d+k} (d+1)^{-k}$$

This also shows that the vectors are independent. Combining the two theorems in section 3.1 we know that we can find a vector \vec{b} in L that satisfies

$$\|\vec{b}\| < (k+d+1)^{\frac{1}{2}} 2^{\frac{k+d+1}{2}} \det(L)^{\frac{1}{k+d+1}}$$

Observe that to get the desired bounds for the c_i 's we need

$$\|\tilde{b}\| < \frac{N}{d+1}$$

A simple calculation shows that to get this we need exactly the bound from the theorem to get this.

To finish the proof we need to prove that we get a nontrivial equation. Since $\|\tilde{b}\| < \frac{N}{d+1}$ we know by the expressions for the last k coordinates that $|s_i| < n_i$. \tilde{b} is also nonzero. This together with the bound for its length imply that there is at least one $s_i \neq 0$. Look at the equation (mod n_i) for the same i . Using that $0 \neq |s_i| < n_i$ and $\gcd(\langle a_{ij} \rangle_{j=0}^d, n_i) = 1$ we see that this is a nontrivial equation.

The proof is complete.

4. Cryptographic Corollaries

We get some immediate corollaries of the main theorem

Corollary 1: Sending linearly related messages using RSA with low exponent is insecure. Sending more than $\frac{d(d+1)}{2}$ messages enables an adversary to recover the messages.

This follows directly from our main theorem assuming that the constants depending on the dimension is small compared to the moduli. In the same spirit we get

Corollary 2: Sending linearly related messages using the Rabin encryption function is insecure. If 4 such messages are sent it is possible to retrieve the message.

If one does a bit of extra work it is possible to say something about the cases of equality ($\frac{d(d+1)}{2}$ and 3 messages respectively) but we omit the details.

Corollary 3: The protocol by Broder and Dolev is insecure if RSA with low exponent is used.

Follows from the analysis in [2] and the main theorem.

The theorem also proves that we should not encode messages that are small known polynomials in some unknown but this seems quite farfetched.

5. Open questions

One interesting open questions is whether we can solve the problem with fewer equations. It does not seem possible to use this line of attack with substantially fewer equation. To see this one might argue as follows:

The probability that $|c_j| < n^{k-j}$ for $j = 1, \dots, d$ for a fixed set of s_i is approximately $n^{-d(d+1)/2}$ and this would indicate that we should have $n^{d(d+1)/2}$ sets of equations to choose between and therefore at least $d(d+1)/2$ equations.

There does not seem to be any way to extend the above attack to RSA with large exponent. The reason being that the integers involved are too big even to write down. There is still a large amount of structure present and it would be interesting to investigate whether this structure could be used.

Acknowledgments: I would like to thank Silvio Micali, Shafi Goldwasser and Benny Chor for suggesting the problem, listening to early solutions and suggesting improvements and simplifications. They also pointed out the flaws in the argument of Broder and Dolev.

References:

- [1] Alexi W., Chor B., Goldreich O. and Schnorr C.P. "RSA/Rabin Bits are $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$ Secure"
FOCS 1984 pp 449-457
- [2] Awerbuch B., Chor B., Goldwasser S. and Micali S. "Provably Secure Coin Flip in a Byzantine Environment", manuscript in preparation.
- [3] Blum M. and Goldwasser S. "An efficient Probabilistic Public Key Encryption Scheme which Hides all Partial Information" Presented in Crypto 1984
- [4] Broder A.Z. and Dolev D. "Flipping Coins in Many Pockets"
FOCS 1984 pp 157-170
- [5] Cassels J.W.S. "Geometry of Numbers" Springer 1959
- [6] Goldwasser S. and Micali S. "Probabilistic Encryption"
JSCC 28 270-299
- [7] Lenstra A.K., Lenstra H.W. and Lovasz L. "Factoring Polynomials with Integer Coefficients" *Mathematische Annalen* 261 (1982) 513-534
- [8] Rivest R.L., Shamir A. and Adleman L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems" CACM 21-2 February 1978.
- [9] Schnorr C.P. "A Hierarchy of Polynomial Basis Reduction Algorithms", manuscript

LENSTRA'S FACTORISATION METHOD BASED ON ELLIPTIC CURVES

N.M.Stephens
University College
Cardiff
Great Britain

0. Introduction

The purpose of this exposition is to explain the method due to H.W.Lenstra, Jr. [1] of determining a non-trivial factor, p , of a composite number, n . The method uses the theory of elliptic curves and has an expected running time of $L(p)^{1/2}$ where $L(p)=\exp(\sqrt{\log p \log \log p})$. The aim of the exposition is to be completely elementary. It begins with an introduction to the arithmetic of elliptic curves sufficient to enable the reader to follow the later section explaining the method. The paper ends with a few remarks on techniques for the practical implementation of the algorithm.

The problem of finding efficient algorithms to decide whether or not a number is prime (primality testing) and to determine a non-trivial divisor of a composite number (factorisation) has a long history and has been considered by many number theorists including Fermat and Gauss. The more recent motivation for the study of the problem is the apparent security of the RSA public key cryptographic system based on the difficulty to factorise a number which is the product of two large primes.

The classical technique of trial division by all numbers up to \sqrt{n} can be used to test the primality of n and to factor n . It has running time $O(\sqrt{n})$ which in terms of $\log n$ (the most reasonable measure of the size of the problem) is $O(\exp(c \log n))$ with $c=1/2$. Thus the trial division technique is said to have exponential running time.

Progress in the factorisation problem was made in the early 1970's. The methods still had exponential running time but the constant c was smaller. Lehman's method was with $c=1/3$; Shanks provably with $c=1/4$ but with $c=1/5$ assuming the extended Riemann Hypothesis; Pollard's rho method has an expected running time with $c=1/4$.

Newer advances were based on Fermat's idea that a non trivial solution of

$$x^2 \equiv y^2 \pmod{n}$$

yields a non-trivial factor of n , viz the highest common factor of $x-y$

and n . The determination of a solution and the analysis of the efficiency of the method required the concept of a "smooth" number (see later). The various methods exploiting this idea are difficult to analyse exactly but may reasonably be assumed to have sub-exponential running times. That is, to factorise n requires time $L(n)^Y$ for some constant Y , and similar storage requirements. The best value of Y so far achieved using these techniques is $Y=1$. In practice, the time to factor $n=pq$, where p and q are primes of size about 10^{30} , on a big machine is 28 hours [2].

1. Elliptic Curves

Let F be a field; for the purpose of this exposition, F may be assumed to be \mathbb{Q} , the rational numbers, or \mathbb{F}_p , the set of residues modulo a large prime p . An elliptic curve is an equation in x and y of the form

$$E: y^2 = x^3 + Ax + B$$

where A, B are integers such that $4A^3 + 27B^2 \neq 0$. (This is a technical condition which prevents the cubic in x having a linear factor squared.)

A point on the curve is either a pair (x, y) with $x, y \in F$ satisfying the equation or the pair $(\infty, \infty) = \underline{0}$. The set of all points $E(F)$ forms an abelian group. The zero of the group is $\underline{0}$. The negative of (x, y) is $(x, -y)$. The sum of $\underline{p}_1 = (x_1, y_1)$ and $\underline{p}_2 = (x_2, y_2)$ with $\underline{p}_1 \neq \underline{p}_2$ and $\underline{p}_1, \underline{p}_2 \neq \underline{0}$ is given as follows:

$$\text{let } m = \begin{cases} (y_2 - y_1) / (x_2 - x_1) & x_1 \neq x_2 \\ (3x_1^2 + A) / (2y_1) & x_1 = x_2 \end{cases}$$

Then $\underline{p}_1 + \underline{p}_2 = \underline{p}_3 = (x_3, y_3)$ is given by

$$x_3 = m^2 - x_1 - x_2$$

$$y_3 = m(x_1 - x_3) - y_1$$

One can check that with addition defined as above that $E(F)$ is indeed an abelian group, but some motivation for the formula would probably make the reader happier and less inclined to skip the rest of this article. The explanation is best illustrated with the field $F = \mathbb{R}$ the set of real numbers. The curve E can then be represented as in figure 1.

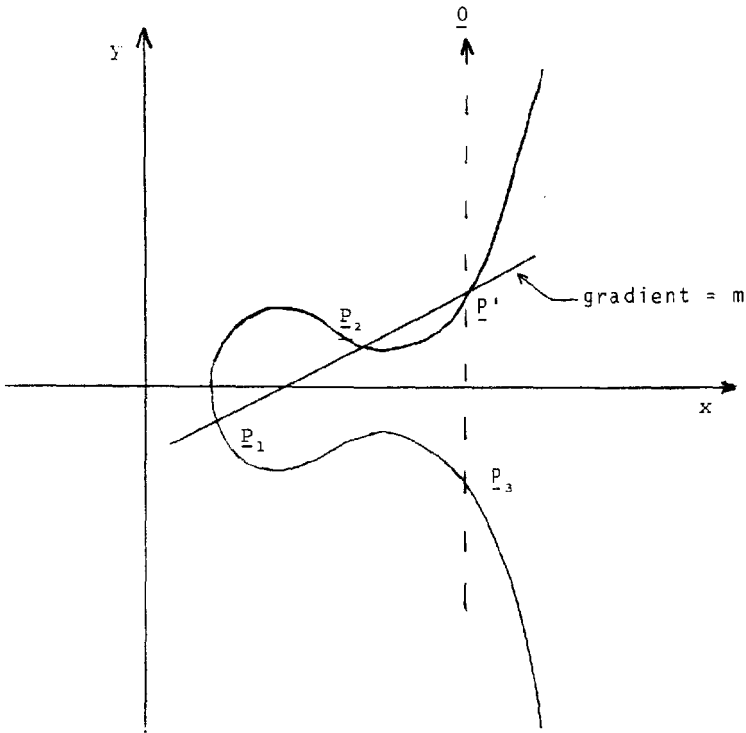


Figure 1

The line joining the points \underline{P}_1 and \underline{P}_2 has gradient m given by the above formula; the alternative is derived from the limiting case when the chord becomes the tangent at \underline{P}_1 . This line intersects the curve at one further point \underline{P}' , whose negative is defined to be the sum of \underline{P}_1 and \underline{P}_2 .

One way of interpreting the addition law on $E(F)$ is to state that the three points \underline{P} , \underline{Q} and $\underline{R} \in E(F)$ are colinear if and only if

$$\underline{P} + \underline{Q} + \underline{R} = \underline{0}$$

Note that the addition law lends itself very easily to computation. If $M\underline{P}$ denotes the point \underline{P} added to itself M times, then $M\underline{P}$ can be computed using the standard divide and conquer technique in $O(\log M)$ arithmetic steps.

Example To illustrate the addition law, consider the elliptic curve

$$E : y^2 = x^3 - 16x + 16 .$$

Over \mathbb{Q} , there is an obvious point $\underline{P} = (0, 4)$. To obtain $2\underline{P} = (0, 4) + (0, 4)$ we need to use the tangent formula for m :

$$m = (3x_1^2 + A)/(2y_1) = -2$$

from which we obtain $2\underline{P} = (4, 4)$. From \underline{P} and $2\underline{P}$ we obtain $3\underline{P}$ by using the chord formula for m :

$$m = (y_2 - y_1)/(x_2 - x_1) = 0$$

and hence $3\underline{P} = (-4, -4)$. Continuing in this way, further multiples of \underline{P} are:

$$4\underline{P} = (8, -20)$$

$$5\underline{P} = (1, -1)$$

$$6\underline{P} = (24, 116)$$

$$7\underline{P} = (-20/9, -388/27)$$

$$8\underline{P} = (84/25, -52/125), \dots$$

The structure of $E(\mathbb{Q})$ In 1922, Mordell showed that $E(\mathbb{Q})$ is a finitely generated abelian group; that is, that all rational solutions on E may be generated using the addition law from a finite number of basic solutions. If T is the subgroup of points of finite order on $E(\mathbb{Q})$ then

$$E(\mathbb{Q}) \cong T \times \mathbb{Z}^g$$

for some finite number g . There is a rich body of theory concerning the structure and computation of $E(\mathbb{Q})$ and many unsolved but important conjectures with far-reaching consequences. These do not concern us here but the interested reader might like to consult [3] for a stimulating account.

Returning to the example above, it can be shown that $E(\mathbb{Q}) \cong \mathbb{Z}$ (i.e. no points of finite order except $\underline{0}$, and $g=1$) and that it is generated by $\underline{P} = (0, 4)$.

We need to state some more facts about an elliptic curve over the finite field F_p . Firstly, the group $E(F_p)$ (which is obviously finite) is either a cyclic group or the product of two cyclic groups.

Secondly, let N_p denote the number of points in $E(F_p)$. Its value will depend on E , that is the values of A and B modulo p . Define a_p by

$$N_p = p+1 - a_p.$$

Since $p+1$ represents the "expected" number of solutions of

$$y^2 \equiv x^3 + Ax + B \pmod{p}$$

including the point at infinity, the value a_p measures the discrepancy of N_p from its expected value. It is known (the Riemann Hypothesis for Abelian Varieties of dimension 1) that

$$-2\sqrt{p} < a_p < 2\sqrt{p}.$$

Moreover, for each integer a in this range, the number of elliptic curves over F_p with $0 \leq A < p$, $0 \leq B < p$ and having $N_p = p+1 - a$ is

$$h(4p-a^2) * (p-1)/2$$

where $h(d)$ is the Hurwitz class number of discriminant $-d$. In particular, to every value a with $|a| < 2\sqrt{p}$, there is a corresponding elliptic curve E over F_p with $a_p = a$.

Example We consider the original example reduced modulo p with $p=5$:

$$E: y^2 = x^3 - x + 1.$$

To determine N_p consider the values of the cubic and thus y for each possible x :

x	0	1	2	3	4	∞
x^3-x+1	1	1	2	0	1	∞
y	± 1	± 1		0	± 1	∞

Thus $N_5 = 8$ and $a_5 = -2$. That $E(F_5)$ is cyclic of order 8 can be clearly seen by reducing \underline{p} , $2\underline{p}$, ..., $8\underline{p}$ from $E(Q)$ to $E(F_5)$ by considering each co-ordinate mod 5. In particular

$$8\underline{p} = (84/25, -52/125) + (\infty, \infty) = \underline{0}$$

2. b-smooth numbers

Let b be a positive integer. A number m is said to be b-smooth if every prime factor q of m is less than or equal to b . Thus

$$m = \prod_{q \leq b} q^{e_q} \quad \text{with} \quad e_q \geq 0.$$

Numbers which are b -smooth are rare. The probability that a random number $m \leq x$ is b -smooth is about u^{-u} where $u = \log x / \log b$. Thus, for example, if $x = 10^{100}$ and $b = 10^{10}$, then $u = 10$ and we see that the probability is 10^{-10} .

3. Pollard's p-1 method

The easiest way to understand Lenstra's factorisation method is to look first at Pollard's so-called $p-1$ method since certain strong

analogies can be drawn.

Suppose that the number n is to be factorised and that it is known that it has a prime divisor p for which $p-1$ is b -smooth. Define M by:

$$M = \prod_{q \leq b} q^{\lfloor \log n / \log q \rfloor}.$$

The exponents of q have been chosen so that we can guarantee that $p-1$ divides M . In particular, for any integer a with $(a, p) = 1$

$$a^{p-1} \equiv 1 \pmod{p}$$

and so

$$a^M \equiv 1 \pmod{p}.$$

To find p then, the method first computes $d \equiv a^M - 1 \pmod{n}$ for some random a - for example $a=2$; secondly it computes the highest common factor of d and n . This will normally be p unless there are other primes q dividing n for which the exponent of $a \pmod{q}$ divides M .

The time for this method is dominated by the time to compute a^M modulo n . Using the divide and conquer technique for evaluation of powers, this can be achieved in $O(\log M)$ modulo n multiplication steps. Hence since $\log M = \sum_{q \leq b} \log q$ the time is $O(b(\log n)^2 / \log b)$.

In view of the earlier remark about the rarity of b -smooth numbers, this method is not worth implementing unless it is known that, for some small b , $p-1$ is b -smooth.

4. Lenstra's method

The crucial results that made Pollard's $p-1$ method successful were that the residues modulo p formed a multiplicative group of order $p-1$, and so $a^{p-1} \equiv 1 \pmod{p}$, and that $p-1$ was b -smooth.

Lenstra's method is based on the same idea. But instead it uses the group (written additively) of points on an elliptic curve E over F_p which has order N_p . It too succeeds whenever N_p is b -smooth. The advantage of the elliptic curve method is that there are a large number of different curves E that can be tried, each with a potentially different value of N_p . For a large number of tries, the hope is that one such N_p is b -smooth for some suitably chosen small number b .

Explicitly, the method is as follows:

- Step 1 Choose a value for b (see later for optimal value) and let $M = b!$ or $M = \text{lcm}(1, 2, \dots, b)$ or some similar large b -smooth number.
- Step 2 Choose an elliptic curve E over \mathbb{Q} at random and a point \underline{P} that satisfies the equation modulo n .
- Step 3 Compute $M\underline{P}$ modulo n using the formulae given in section 1. If this computation fails to give a factor of n go back to Step 2 and choose a different curve E .

How does the computation of $M\underline{P}$ modulo n yield a factor of n ? The addition formulae supplied were for computation over a field. The formula for m , the gradient, involves one division; now over a field, this causes no problem because if the denominator is zero then the required sum is the point $\underline{Q} = (\infty, \infty)$. Modulo a composite number n , however, the division is only possible if the denominator d is co-prime to n . So the computation "fails" whenever $(d, n) > 1$. In particular, it fails when p is a divisor of n and N_p divides M . This failure however gives a factorisation of n and signifies, in fact, the success of the method.

The method then successfully factors n when the value of N_p for the chosen elliptic curve is a b -smooth number dividing M and there is a q dividing n such that the order of \underline{P} in $E(\mathbb{F}_q)$ does not divide M .

Example The method is illustrated by attempting to factorise $n=187$ with $b=3$, $M=3!=6$ and $\underline{P}=(0,5)$ on the curve

$$E: y^2 = x^3 + x + 25.$$

To compute $\underline{P} + \underline{P}$: $M = 1/10 \equiv -56 \pmod{187}$ and $2\underline{P} = (-43, 18)$.

To compute $2\underline{P} + 2\underline{P}$: $M = (-62)/36 \equiv 71 \pmod{187}$ and $4\underline{P} = (78, -7)$.

To compute $2\underline{P} + 4\underline{P}$: $M = (-25)/(-66)$.

But $(-66, 187) = 11$; thus in three steps, the method produces the factor 11 of 187.

5. Concluding Remarks

The following theorem is due to Lenstra. The proof makes one "reasonable" assumption about the number of b -smooth numbers in a small interval.

Theorem The elliptic curve method with $b = L(p)^{1/\sqrt{2}}$ splits any integer n in expected time $O(L(p)^{\sqrt{2}+o(1)} (\log n)^2)$ where p is the smallest prime divisor of n .

Corollary The elliptic curve method can be used to factor completely any n in expected time $O(L(n)^{1+o(1)})$.

These results show that asymptotically, Lenstra's method is as good as any previously known method to factorise n . It has, however, in addition several important advantages. Firstly, it is easy to program, a non-sophisticated version requiring about 100 lines of code. Secondly, it requires very little storage and can be conveniently run as a background job. Thirdly, if n has a small prime factor, this can be expected to be found sooner than larger ones and helps to terminate the method in a shorter time. The previous fast methods operated in time which is independent of the size of the factors. Fourthly, it is ideally suited to implementation in parallel; many elliptic curves can be tried simultaneously and independently.

In practice, with the implementations so far, the method is not as fast as existing methods when applied to composite numbers which are the product of two large primes. To the author's knowledge the best results using Lenstra's method have been achieved by P. Montgomery [4] who has for example successfully factored the 74 digit number

$$(5^{106} + 1)/2$$

into two factors, one of which is approximately 10^{22} .

There are various implementation techniques to speed up the method of factorisation described in this article. Many of these techniques are improved by practical experimentation and it is the author's view that the method has not been around long enough to achieve its best potential. One of the crucial problems in increasing the speed of the method is to avoid too many inversions modulo n . Another is to make a sensible choice of b and of M . Clearly small primes in M need to have a higher exponent than big primes. Experiments indicate that whereas $M=b!$ has the exponents of small primes too big, $M=\text{lcm}(1, \dots, b)$ has the exponents too small.

6. References

1. H.W.Lenstra Jr. Elliptic Curve Factorisation and Primality Testing. Paper to Computational Number Theory Conference at Arcata, California. August 1985.
2. R. Silverman, Paper to Computational Number Theory Conference at Arcata, California. August 1985.
3. N.Koblitz. Introduction to Elliptic Curves and Modular Forms. Springer-Verlag 1984.
4. P.L.Montgomery. Experiences using Elliptic Curve Method of Factorisation. Paper to Computational Number Theory Conference at Arcata, California. August 1985.

Use of Elliptic Curves in Cryptography

Victor S. Miller

Exploratory Computer Science, IBM Research, P.O. Box 218, Yorktown Heights, NY 10598

ABSTRACT

We discuss the use of elliptic curves in cryptography. In particular, we propose an analogue of the Diffie-Hellmann key exchange protocol which appears to be immune from attacks of the style of Western, Miller, and Adleman. With the current bounds for infeasible attack, it appears to be about 20% faster than the Diffie-Hellmann scheme over $GF(p)$. As computational power grows, this disparity should get rapidly bigger.

INTRODUCTION

Elliptic curves have been objects of intense study in Number Theory for the last 90 years. To quote Lang "It is possible to write endlessly on Elliptic Curves (This is not a threat)." [1]. Recently [2], H.W. Lenstra has proposed a new integer factorization algorithm based on the arithmetic of elliptic curves, which, under reasonable hypotheses, runs at least as fast as the best known factorization algorithm, and uses a negligible amount of storage. This has obvious implications for cryptographic techniques depending on the difficulty of factoring. It is my intent to show that elliptic curves have a rich enough arithmetic structure so that they will provide a fertile ground for planting the seeds of cryptography.

NOTATION AND RESUME OF PROPERTIES OF ELLIPTIC CURVES

If S is a finite set, we denote its cardinality by $|S|$. If p is a prime number, and $n \neq 0$ is an integer, we denote by $v_p(n)$ the exact exponent of p dividing n . If $a = b/c$ is rational, then we set $v_p(a) = v_p(b) - v_p(c)$. As usual, \mathbb{Q} denotes the rational numbers, and \mathbb{Z} denote the integers. If $n \neq 0$ is an integer, let $\mathbb{Q}_{(n)}$ denotes the subring of \mathbb{Q} consisting of elements whose denominators are relatively prime to n . If σ denotes a set of primes then let \mathbb{Z}_σ denote those rational numbers whose denominators are divisible only by primes in σ . Note that if no prime in σ divides n , then \mathbb{Z}_σ is a subring of $\mathbb{Q}_{(n)}$.

An (affine) algebraic group defined over a ring R is a set of simultaneous polynomial equations in x_1, \dots, x_n , with coefficients in R :

$$f_1(P) = 0, \dots, f_t(P) = 0$$

along with a composition law, and inverse given by n polynomial functions with coefficients in R .

$$\begin{aligned} m_i(x_1, \dots, x_n, y_1, \dots, y_n) \\ a_i(x_1, \dots, x_n) \end{aligned}$$

which satisfies the usual axioms for a group. If G is an algebraic group, and S is a ring which has a multiplication by elements of R defined, then $G(S)$ denotes the set of solutions to the polynomial equations with the variables having values in S . The law of composition given above then makes $G(S)$ into a group. We also may have a projective algebraic group with the same definition as above, except that the polynomials must be homogeneous of the same degree. In this case $G(S)$ denotes the set of solutions all of whose coordinates are not zero, with two solutions being considered the same if one is a scalar multiple of the other. Note that in this case, the law of composition really consists of a set of rational functions.

As an example, we have the multiplicative group:

$$G_m: xy = 1$$

with law of composition $((x_1, y_1), (x_2, y_2)) \rightarrow (x_1 x_2, y_1 y_2)$ and inverse $(x, y) \rightarrow (y, x)$.

Define the logarithmic height of a point: Given a point $P = (x_1, x_2, \dots, x_n)$ with rational coordinates, let D be a common denominator for all the x_i such that, there is a j such that $(Dx_j, D) = 1$. The logarithmic height, $h(P) = \log \max(|D|, |Dx_1|, \dots, |Dx_n|)$. This height is a measure of the number of bits needed to write down the point P . Let $H_n(K) = \{P \in \mathbb{Q}^n \mid h(P) \leq K\}$.

Let p_i denote the i -th prime number, and $G_m(r)$ be the subgroup of $G_m(\mathbb{Q})$ generated by p_1, \dots, p_r . Note that $G_m(r)$ is the same as $G_m(\mathbb{Z}_\sigma)$ where σ consists of the set of primes p_1, \dots, p_r . Also let $G_m(r, K) = G_m(r) \cap H_2(K)$.

An elliptic curve defined over a field F is the curve defined by the following equation:

$$E: y^2 = x^3 + ax + b$$

where a and b are elements of F (assumed to have characteristic $\neq 2$ or 3 . There is a slightly more complicated formulation in those cases.). There is a natural law of composition on the points of E obtained by the "tangent and chord method": Given two points P and Q , the straight line containing them intersects the curve in a third point R (if $P = Q$ take the tangent to the curve at P). Define $P + Q$ as being the point $(x(R), -y(R))$. This provides a commutative and associative law of composition, whose zero element is the point at infinity: (∞, ∞) . We denote the set of points (including the point at infinity) of the curve E with coordinates in the field F by $E(F)$. The discriminant of the curve $\Delta = 16(4a^3 - 27b^2)$. The elliptic curve

$$E_\lambda: y^2 = x^3 + \lambda^4 ax + \lambda^6 b$$

is isomorphic to the curve E above by the substitution

$$(x, y) \rightarrow \left(\frac{x}{\lambda^2}, \frac{y}{\lambda^3}\right)$$

We say the E is minimal if a and b are integers, and there is no integer $\lambda \neq \pm 1$ such that $\lambda^4 \mid a$ and $\lambda^6 \mid b$. Clearly, every elliptic curve is isomorphic to a minimal one. We denote the discriminant of the minimal curve isomorphic to E by Δ_{\min} . There is a slightly more general definition of minimal by using a more complicated model for an elliptic curve (see [1]). Its value of Δ_{\min} differs by a factor dividing 24, from the one described above.

To calculate multiples of a point $P = (x, y)$ we may use the following recurrences (see Lang [1], p. 37):

$$\begin{aligned}
g_1 &= 1 \\
g_2 &= 1 \\
g_3 &= 3x^4 + 6ax^2 + 12bx - a^2 \\
g_4 &= x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3 \\
g_{2n} &= g_n(g_{n+2}g_{n-1}^2 - g_{n-2}g_{n+1}^2) \\
g_{4n+1} &= 16y^4g_{2n+2}g_{2n}^3 - g_{2n+1}^3g_{2n-1} \\
g_{4n+3} &= g_{2n+3}g_{2n+1}^3 - 16y^4g_{2n+2}^3g_{2n} \\
f_{2n} &= 2yg_{2n} \\
f_{2n+1} &= g_{2n+1} \\
\phi_n &= xf_n^2 - f_{n+1}f_{n-1} \\
4y\omega_n &= f_{n+2}f_{n-1}^2 - f_{n-2}f_{n+1}^2
\end{aligned}$$

Then

$$n(x, y) = \left(\frac{\phi_n}{f_n^2}, \frac{\omega_n}{f_n^3} \right)$$

Using the above recursions we may calculate the coordinates of the above point in $26 \log_2 n$ multiplications.

We let F_q denote $\text{GF}(q)$, the finite field with q elements. We now state a few results for elliptic curves which are needed for the discussion in the next section. Two good general references for elliptic curves are Cassels [11] and Lang [1]. All of the results quoted below are contained therein, unless indicated otherwise. The number of points, $|E(F_p)| = p + 1 - a_p$ where $|a_p| \leq 2\sqrt{p}$ (the "Riemann hypothesis for finite fields" proved by Hasse in 1931 for Elliptic curves). The Mordell-Weil theorem states that the rank of the free part of the group $E(Q)$ is finite (for any specific E). In fact it is usually quite small. Indeed, no one to date has been able to find an elliptic curve with rational coefficients whose rank is greater than 14 (this record is held by J-F Mestre, see [12] for a description of a rank 12 case).

A fundamental theorem of Neron and Tate (see [1]) is that there exists a unique positive semi-definite quadratic function $\hat{h}(P)$ such that for all $P \in E(Q)$ (even on $E(M)$ where M is a number field) such that

$$h(P) = \hat{h}(P) + O(1)$$

The $O(1)$ is quite small, even being bounded by $\log \max(|a|, |b|)$ (see Zimmer [15]). In fact this bound always seems to be much too large (see [16]). We also have $\hat{h}(P) = 0$ if and only if P is a

point of finite order (of which there are at most 16, by a theorem of Mazur). The meaning of $\hat{h}(P)$ being a quadratic function is that

$$\langle P, Q \rangle = \hat{h}(P + Q) - \hat{h}(P) - \hat{h}(Q)$$

is a positive definite inner product. If P_1, \dots, P_r is a basis for the points of $E(\mathbb{Q})$ of infinite order, we define the regulator to be

$$R = \det(\langle P_i, P_j \rangle)$$

This value is independent of the basis chosen. We also define $|P| = \sqrt{\langle P, P \rangle}$. In this case $\langle P, P \rangle = 1/2\hat{h}(P)$.

KEY EXCHANGE, AND DISCRETE ELLIPTIC LOGARITHMS

The Diffie-Hellman key exchange protocol [3] was proposed to allow the agreement on a secret key between two parties communicating over an insecure channel. It operates as follows: A large prime p and a primitive root g of p are made public. Party A chooses an exponent a between 0 and $p - 1$ at random. Party B does the same with an exponent b . Party A transmits g^a to B, and vice-versa. Both parties agree on g^{ab} . The security of this protocol rests on two unproven (but reasonable) assumptions:

1. Any method of obtaining g^{ab} from g^a and g^b would be as hard as obtaining a from g^a (taking "discrete logarithms").
2. If $p - 1$ did not have only small prime factors, that finding discrete logarithms was intractible (i.e. could not run in time polynomial in $\log p$).

Neither assumption has been disproven. However, Western and Miller [4], and Adleman [5] have come up with algorithms for the discrete logarithm problem which run in time $L(p)$, where

$$L(x) = \exp(\sqrt{\log x \log \log x})$$

In addition, Pohlig and Hellman [17], and Pollard [18] have a method for calculating discrete logarithms, depending only on the fact that we are working in a group, which runs in time $O(\sqrt{p'})$ where p' is the largest prime factor of $p - 1$. Given current processing speeds, this escalates the size of the prime p which must be used, in order to make this method more secure. A figure of $p \approx 2^{512}$ seems to be necessary.

The above protocol really only uses the property that we are working in a group. As stated above, the points on an elliptic curve have the structure of an abelian group. Thus we may make the analogous constructions over elliptic curves. We shall briefly describe the "Index Calculus" algorithm of Adleman, and Western and Miller, and give arguments why such an algorithm is not likely to work on elliptic curves. We have the reduction map:

$$G_m(Q(p)) \rightarrow G_m(F_p)$$

denoted by $x \rightarrow \bar{x}$. Now

$$|G_m(r, \log \sqrt{p}/2)| = |\overline{G_m(r, \log \sqrt{p}/2)}|$$

Let $\text{prob}(r) = |G_m(r, \log \sqrt{p}/2)| / (p-1)$. This is the probability that an element of the multiplicative group is in the above image. As r increases to $\pi(\sqrt{p}/2)$, $\text{prob}(r)$ increases to 1.

The "index calculus" method fixes a value of r , and chooses elements $a \in F_p$ at random until there is $x \in G_m(r, \log \sqrt{p}/2)$ such that $\bar{x} = g^a$. The probability of that succeeding is $\text{prob}(r)$. To each such successful test we have an equation

$$a = \nu_0 l_0 + \dots + \nu_r l_r \bmod p-1 \quad (1)$$

where

$$x = (-1)^{\nu_0} p_1^{\nu_1} \dots p_r^{\nu_r} \quad (2)$$

and l_k is such that $p_k = g^{l_k} \bmod p-1$ where $p_0 = -1$. Evidently, we have $l_0 = (p-1)/2$. We need to generate r such independent equations. Once we have accumulated them, we may solve for the l_i .

Given $z \in F_p$, we find l such that $z = g^l$ as follows: Choose $a \bmod p-1$ at random until there exists $x \in G_m(r, \log \sqrt{p}/2)$ such that $\bar{x} = z^a$. Then

$$al = \nu_0 l_0 + \nu_1 l_1 + \dots + \nu_r l_r$$

where

$$x = p_0^{\nu_0} p_1^{\nu_1} \dots p_r^{\nu_r}$$

We may then solve for l . Each such test has probability $\text{prob}(r)$ of succeeding.

There is a trade-off between increasing r in order to make $\text{prob}(r)$ bigger (in order to decrease the expected number of tests to make), and in decreasing r in order to make the calculation of the decomposition (2) faster, and of solving a smaller system of equations. Fortunately, good algorithms exist for both the latter problems. Using the new factorization algorithm of Lenstra [2], we may find the decomposition (2), or signal failure, in time

$$O(L(p)^{\sqrt{2a} + \epsilon})$$

where

$$\alpha = \frac{\log p_r}{\log p} \approx \frac{\log r + \log \log r}{\log p}$$

The equations (1) are provably sparse, namely at most $\log p$ of the v_i are $\neq 0$, because $p_1 \dots p_r \approx e^r$ by the prime number theorem. We may solve these equations in random time $O(r^2 \log^2 p)$ by the algorithm of Wiedemann [13]. It is evident that this last figure is the big bottleneck in trying to make r larger. It turns out that the optimum trade off is made by letting $r = L(p)^{c/2}$ for some small constant c between $3/2$ and 2 . The total running time turns out to be $L(p)^c$. Recently, Coppersmith, Odlyzko, and Schroepel have devised a slightly more complicated variant of the above, which has the above running time with $c = 1$.

The reason why the above algorithm works so well, is that there are lots of free generators for the group $G_m(\mathbb{Q}_{(p)})$, which have fairly small heights. If one tries to use an analogous method with elliptic curves, one immediately runs into the barrier of the Mordell-Weil Theorem (see above). We show below, that this finitude of the rank combined with other estimates, that it is extremely unlikely that an "index calculus" attack on the elliptic curve method will ever be able to work. We may view $E(\mathbb{Q}) \otimes \mathbb{R}$ as an r -dimensional inner product space, with the inner product given above, which contains $E(\mathbb{Q})$ as a lattice, whose fundamental domain has volume \sqrt{R} . Thus,

$$|E(\mathbb{Q}) \cap H(K)| = 2 \frac{V_r}{\sqrt{R}} K^{r/2} + O(K^{(r-1)/2})$$

where V_r is the volume of the r dimensional sphere of radius 1. Thus, unless the rank of the curve can be made very large, and the regulator made fairly small, the probability of a point of $E(\mathbb{F}_p)$ lifting to a point on $E(\mathbb{Q})$ whose height is bounded by something reasonable (say a polynomial in $\log p$) is vanishingly small. In particular, in order to make the probability of finding a point with a specified height $< p^a$ it is necessary to make $K = \Omega(p^{(1-a)/r})$. That is we must compute points whose coordinates are represented by rational numbers whose length is exponential in $\log p$. That is rather a daunting prospect!

Despite the remarks above about it being difficult to find curves of large rank, it is widely believed that there is no bound on the rank attainable. However, it is also true that $\text{rank}(E(\mathbb{Q})) = O(\log \max(|a|, |b|))$. This shows that the size of the coefficients needs to be exponentially larger than the rank. This would seem to preclude high rank from the point of view of computational complexity. In fact, the above bound is really quite bad, which would tend to make the situation much worse from the point of view of computational complexity. As far as a lower bound on the regulator, Lang has conjectured [1], and Silverman proved [7] (in some cases)

that if $\hat{h}(P) \neq 0$ that $\hat{h}(P) > c_1 \log |\Delta_{\min}| + c_2$ for some constants c_i . This estimate is even true over algebraic number fields, with the constants depending on the field. Laurent [8] gives a precise lower bound for the constant c_1 , if one has $c_2 = 0$ (this only make c_1 larger), of $c_3/(D(\log \log D)^3)$ where D is the degree of the field above \mathbb{Q} and c_3 is an absolute constant independent of the curve and the field. These estimates say that the regulator can't be too small, as long as a and b can't get too big. This remark would seem to preclude an attack which tries to look at points in $E(M)$ for some finite field extension M of \mathbb{Q} .

Even if one could somehow get around the barrier mentioned above there is still the problem of actually lifting a point. In the original case of G_m it is trivial, or nearly so. In the case of an elliptic curve it seems to be much more difficult. If we are given a point $(x, y) \in E(\mathbb{F}_p)$ and some point $(x_1, y_1) \in E(\mathbb{Z}/p^k\mathbb{Z})$ which projects to the original point, we could find a rational point (X, Y) whose height is bounded by $k \log p - \log 8$ by an integer basis reduction algorithm (L^3 or Kannan) in the 3 dimensional lattice generated by the vectors

$$\begin{pmatrix} 1 \\ x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} 0 \\ p^k \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ p^k \end{pmatrix}$$

However, there are many possible choices for (x_1, y_1) , about $p^{k/3}$ of them. Furthermore, even though they are parametrizable, the parametrization is non-linear. Thus, unless there is a new idea, it would seem that this is another barrier, difficult to surmount.

IMPLEMENTATION AND PRACTICE

A number of details need to be addressed in order to make this scheme practical:

1. The actual algorithm for multiplication on an elliptic curve
2. The choice of the parameters A and B for the elliptic curve.
3. The choice of the prime modulus p .
4. What information needs to be transmitted.

There are two possible algorithms that one could use for multiplying a point by an integer: the recursion cited above, or repeated use of addition and doubling with the binary method for multiplication. In either algorithm, it appears to be best to represent the points on the curve in the following form: Each point is represented by the triple (x, y, z) which corresponds to the point $(x/z^2, y/z^3)$. This is a homogeneous representation with x having weight 2, y having weight 3, and z having weight 1. If this representation is used with the recursions in the first section, then it is easily checked that the only change is in the initialization. A simple induction shows that g_{2n} has weight $4n^2 - 4$, and that g_{2n+1} has weight $4n^2 + 4n$.

In order to be secure from the Pohlig-Hellmann (or Pollard) algorithm, it is necessary that N_p , the number of points of E in F_p , have a prime factor $> p^\alpha$, for α as close to 1 as possible. This is made possible by the algorithm of Schoof [19], which calculates N_p in time polynomial in $\log p$. In general it is not hard to find such good p . Theoretically, the best result known is one of Fouvry [20]: For any fixed non-zero integer a , a positive proportion of primes p have the property that the largest prime factor of $p + a$ is $\geq p^\delta$ where $\delta = 0.6687$.

Instead of using the Schoof algorithm, when searching for a good p , I have taken the following approach: Choose the curve to be:

$$E: y^2 = x^3 - ax$$

where a is not a perfect square. This curve has complex multiplication by $\sqrt{-1}$, and there is an exact formula for N_p (see [10]). In the case $p \equiv 3 \pmod{4}$ we have $N_p = p + 1$. This is the so-called "supersingular" case. In this case we know even more. It is well known (see [1]) that any field containing the coordinates of all points of order l also contains the l -th roots of unity. This shows that a necessary condition for group of point over F_p to contain a subgroup isomorphic to $\mathbb{Z}/l\mathbb{Z} \times \mathbb{Z}/l\mathbb{Z}$ is that $l \mid p - 1$. Because the number of points in the supersingular case is $p + 1$ we have 2 as the only possibility for l . But, in our case, this happens if and only if, a is a quadratic residue modulo p . To sum up, in the case above the group of points modulo p is of order $p + 1$, cyclic in the case $(a/p) = -1$, and a product of a cyclic group of order 2 and a cyclic group of order $(p + 1)/2$ when $(a/p) = 1$.

The above choice of curve was taken for convenience in calculation. However, it may be prudent to avoid curves with complex multiplication because the extra structure of these curves might somehow be used to give a better algorithm.

Finally, it should be remarked, that even though we have phrased everything in terms of points on an elliptic curve, that, for the key exchange protocol (and other uses as one-way functions), that only the x -coordinate needs to be transmitted. The formulas for multiples of a point cited in the first section make it clear that the x -coordinate of a multiple depends only on the x -coordinate of the original point.

BIBLIOGRAPHY [1] Lang, Serge, *Elliptic Curves: Diophantine Analysis*, Springer-Verlag New York, 1978.

[2] Lenstra, H. W., Letter to A. M. Odlyzko.

[3] Diffie, W. and Hellman M., *New Directions in Cryptography*, IEEE Trans. Inform. Theory, IT-22 (1976), 644-654.

[4] Western, A. E., and Miller, J. C. P., *Table of Indices and Primitive Roots*, Royal Society Mathematical Tables, vol. 9, Cambridge Univ. Press, 1968.

- [5] Adleman, L., A subexponential algorithm for the discrete logarithm problem with applications to cryptography, Proc. 20th IEEE Found. Comp. Sci. Symp. (1979), 55-60.
- [6] Odlyzko, A. M., Discrete logarithms in finite fields and their cryptographic significance, preprint.
- [7] Silverman, J., Lower bound for the canonical height on elliptic curves, Duke Math. J. 48, 633-648 (1981).
- [8] Laurent, M., Minoration de la hauteur de Neron-Tate, Seminaire de Theorie des Nombres, Paris 1981-82, 137-151, Birkhauser (1983).
- [9] Birch, B. J., Swinnerton-Dyer H.P.F., Notes on Elliptic Curves I, J. reine u. angewandte Math., 212, 7-25 (1963).
- [10] Birch, B. J., Swinnerton-Dyer H.P.F., Notes on Elliptic Curves II, J. reine u. angewandte Math., 218, 79-108 (1965).
- [11] Cassels, J. W. S., Diophantine Equations with special reference to elliptic curves, J. London Math. Soc., 41, 193-291 (1966).
- [12] Mestre, J-F., Courbes elliptique et formule explicites, Seminaire de Theorie des Nombres, Paris 1981-82, 179-187, Birkhauser (1983).
- [13] Wiedemann, D., Solving sparse linear equations over finite fields, preprint.
- [14] Coppersmith, D., Odlyzko, A. M., and Schroepel, R., Discrete logarithms in $GF(p)$, IBM Research Report RC 10985 (1985).
- [15] Zimmer, H. G., On the difference of the Weil height and the Neron-Tate height, Math. Z. 147 (1976) 35-51.
- [16] Buhler, J., Gross, B., and Zagier, D., On the conjecture of Birch and Swinnerton-Dyer for an elliptic curve of rank 3, preprint.
- [17] Pohlig, S. and Hellman, M., An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance, IEEE Inform. Theory IT-24 (1978), 106-110.
- [18] Pollard, J. M., Monte Carlo methods for index computation (mod p), Math. Comp. 32 (1978), 918-924.
- [19] Schoof, R., Elliptic Curves over finite fields and the computation of square roots mod p , Report 83-09, Math. Inst. Univ. v. Amsterdam (1983).
- [20] Fouvry, E., Theoreme de Brun-Titchmarsh; application au theoreme de Fermat, Invent. Math. 79 (1985), 383-407.
- [21] Bremner, A. and Cassels, J. W. S., On the Equation $Y^2 = X(X^2 + p)$, Math. Comp. 42 (1984), 257-264.

Cryptography with Cellular Automata

Stephen Wolfram

The Institute for Advanced Study, Princeton NJ 08540.

(November 1985)

EXTENDED ABSTRACT*

This abstract discusses a stream cipher based on a simple one-dimensional cellular automaton. The cellular automaton consists of a circular register with N cells, each having a value a_i equal to 0 or 1. The values are updated synchronously in discrete time steps according to the rule

$$a'_i = a_{i-1} \text{ XOR } (a_i \text{ OR } a_{i+1}) \quad , \quad (1a)$$

or, equivalently,

$$a'_i = (a_{i-1} + a_i + a_{i+1} + a_i a_{i+1}) \bmod 2 \quad . \quad (1b)$$

The initial state of the register is used as a seed or key. The values $a^{(t)}$ attained by a particular cell through time can then serve as a random sequence. Ciphertext C can be obtained from binary plaintext P as usual according to $C_i = P_i \text{ XOR } a^{(t)}$; the plaintext can be recovered by repeating the same operation, but only if the sequence $a^{(t)}$ is known.

Cellular automata such as (1) have been investigated in studies of the origins of randomness in physical systems [2]. They are related to non-linear feedback shift registers, but have slightly different boundary conditions.

Figure 1 shows the pattern of cell values produced by (1) with a seed consisting of a single nonzero cell in a large register. The time sequence of values of the centre cell shows no statistical regularities under the tests of ref. [3] (for sequence lengths up to $2^{19} \approx 5 \times 10^5$). Some definite spacetime patterns are nevertheless produced by the cellular automaton rule.

In the limit $N \rightarrow \infty$, the cellular automaton evolution is like an iterated continuous mapping of the Cantor set, and can be studied using dynamical systems theory [4]. One result is that the evolution is unstable with respect to small perturbations in the initial seed. A change produced by reversing a single cell value typically expands at a rate given by Lyapunov exponents, equal to 0.25 on the left, and 1

* Many more details are given in ref. [1].

on the right. Length T time sequences of cell values are found however to be affected on average only by about $1.19T$ initial values.

Iterations of the cellular automaton rule (1) can be considered as Boolean functions of initial cell values. Disjunctive normal forms (minimized using [5]) for these functions are found to increase in size roughly as $4^{0.65t}$, giving some indication of the complexity of the cellular automaton evolution.

Figure 2 shows the complete state transition diagram for the cellular automaton (1) in a register of size $N=11$. For large N , an overwhelming fraction of states lie on the longest cycle. But there are also shorter cycles, often corresponding to states with special symmetries. Figure 3 shows the length of the longest cycle as a function of N . The results (up to $N=53$, which gives cycle length 40114679273) fit approximately $2^{0.61N}$. The mapping (1) is not a bijection, but is almost so; only a fraction $(\kappa/2)^N \approx 0.85^N$ of states do not have unique predecessors [6] (κ is the real root of $4\kappa^3 - 2\kappa^2 - 1 = 0$).

The security of a cryptographic system based on (1) relies on the difficulty of finding the seed from a time sequence of cell values. This problem is in the class NP. No systematic algorithm for its solution is currently known that takes a time less than exponential in N . No statistical regularities have been found in sequences shorter than the cycle length.

One approach to the problem of finding the seed [6] uses the near linearity of the rule (1). Equation (1) can be written in the alternative form $a_{i+1} = a_i' \text{ XOR } (a_i \text{ OR } a_{i+1})$. Given the values of cells in two adjacent columns, this allows the values of all cells in a triangle to the left to be reconstructed. But the sequence provided gives only one column. Values in the other column can be guessed, and then determined from the consistency of Boolean equations for the seed. But in disjunctive normal form the number of terms in these equations increases linearly with N , presumably making their solution take a time more than polynomial in N .

The cellular automaton (1) can be implemented efficiently on an integrated circuit; it requires less than ten gate delay times to generate each output bit, and can thus potentially be used in a variety of high-bandwidth cryptographic applications.

Much of the work summarized here was done while I was consulting at Thinking Machines Corporation (Cambridge, MA). I am grateful for discussions with many people, including Persi Diaconis, Carl Feynman, Richard Feynman, Shafi Goldwasser, Erica Jen and John Milnor.

References

1. S. Wolfram, "Random sequence generation by cellular automata", to be published in *Advances in Applied Mathematics*.
2. S. Wolfram, "Origins of randomness in physical systems", *Phys. Rev. Lett.* **55**, 449 (1985); S. Wolfram, "Cellular automata as models of complexity", *Nature* **311**, 419 (1984).
3. D. Knuth, *Seminumerical Algorithms*, (Addison-Wesley, 1981).
4. S. Wolfram, "Universality and complexity in cellular automata", *Physica* **10D**, 1 (1984).
5. R. Rudell, *espresso* software program, Computer Science Dept., University of California, Berkeley (1985).
6. C. Feynman and R. Feynman, private communication.

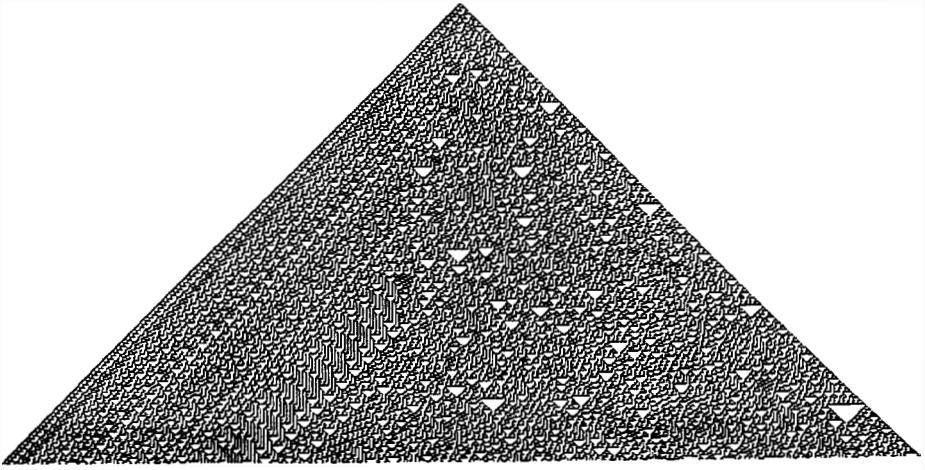


Figure 1. Pattern produced by evolution according the cellular automaton of eqn. (1) from a simple seed containing a single nonzero bit. 250 successive states of an arbitrarily large register are shown; black squares represent nonzero cells. Columns of cell values, say in the centre, seem random for practical purposes.

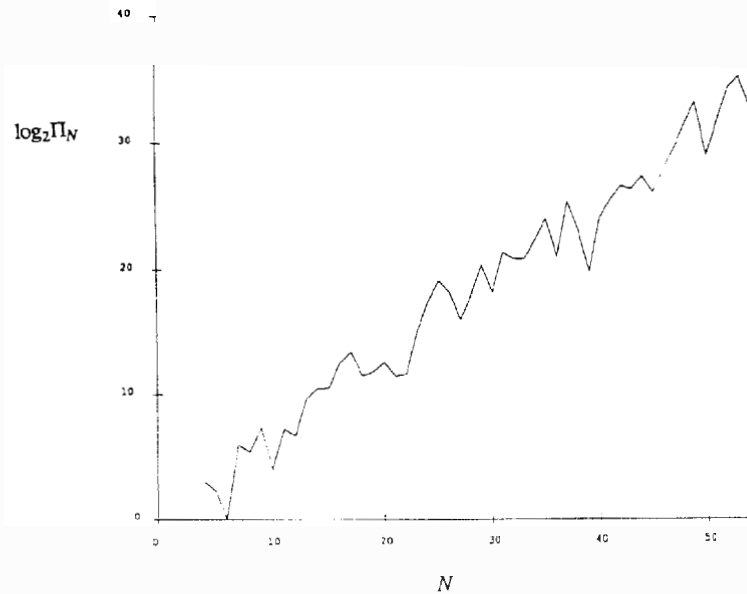


Figure 3. Length Π_N of the longest cycle as a function of register size N .

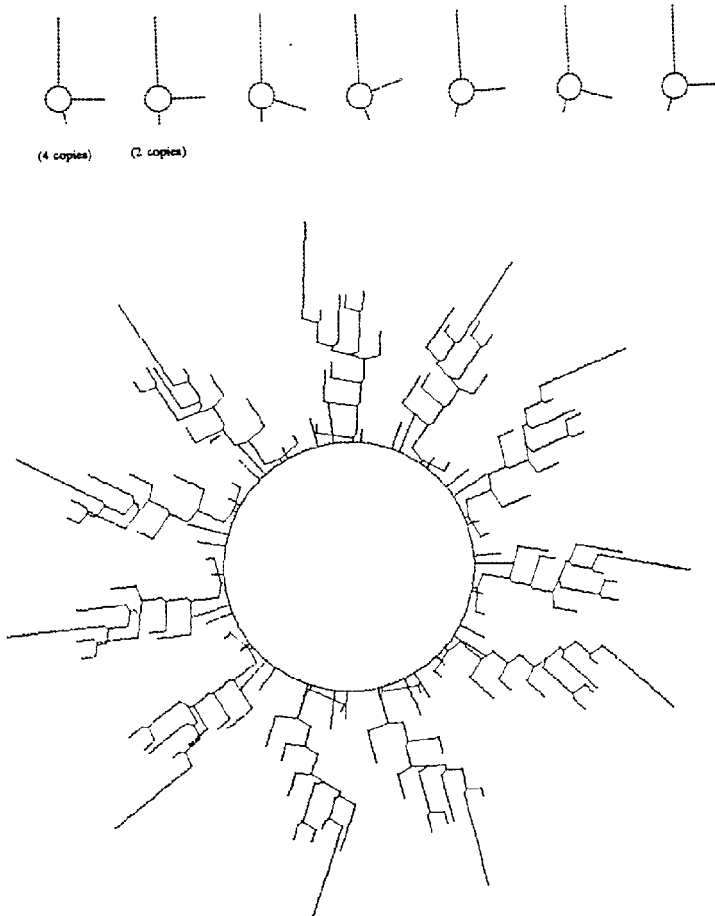


Figure 2. Complete state transition diagram for the cellular automaton of eqn. (1) in a circular register of size $N=11$. There are 2^N states, each represented by dots. Evolution from any state leads eventually to one of the cycles shown.

Efficient Parallel Pseudo-Random Number Generation

J. H. Reif¹

J. D. Tygar²

Aiken Computation Laboratory

Harvard University

Cambridge, MA 02138

0. Abstract

We present a parallel algorithm for pseudo-random number generation. Given a seed of n^ϵ truly random bits for any $\epsilon > 0$, our algorithm generates n^c pseudo-random bits for any $c > 1$. This takes poly-log time using $n^{\epsilon'}$ processors where $\epsilon' = k\epsilon$ for some fixed small constant $k > 1$. We show that the pseudo-random bits output by our algorithm can not be distinguished from truly random bits in parallel poly-log time using a polynomial number of processors with probability $1/2 + 1/n^{O(1)}$ if the multiplicative inverse problem almost always can not be solved in **RNC**. The proof is interesting and is quite different from previous proofs for sequential pseudo-random number generators.

Our generator is fast and its output is provably as effective for **RNC** algorithms as truly random bits. Our generator passes all the statistical tests in KNUTH[14].

Moreover, the existence of our generator has a number of central consequences for complexity theory. Given a randomized parallel algorithm \mathcal{A} (over a wide class of machine models such as parallel RAMs and fixed connection networks) with time bound $T(n)$ and processor bound $P(n)$, we show \mathcal{A} can be simulated by a parallel algorithm with time bound $T(n) + O((\log n)(\log \log n))$, processor bound $P(n)n^{\epsilon'}$, and only using n^ϵ truly random bits for any $\epsilon > 0$.

¹Supported in part by NSF grant NSF-MCS-79-21024 and ONR contract N0014-80-C-0674.

²Supported in part by a NSF graduate fellowship and NSF grant MCS-81-21431.

Also, we show that if the multiplicative inverse problem is almost always not in RNC, then RNC is within the class of languages accepted by uniform poly-log depth circuits with unbounded fan-in and strictly sub-exponential size $\bigcap_{\epsilon > 0} 2^{n^\epsilon}$.

1. Introduction

A number of parallel randomized algorithms have appeared recently. These algorithms typically use a large number of random bits which must be generated in a small amount of time. Nonetheless, the area of parallel random bit generation remains unexplored.

In reality, our computers are deterministic and unable to generate truly random values. But we can give algorithms which will give pseudo-random bits on input of a random seed s_0 . These pseudo-random bits satisfy conditions which suggest that for algorithmic purposes they are as effective as truly random bits.

What conditions should a pseudo-random bit sequence satisfy?

Improving an idea by SHAMIR[17], BLUM-MICALI[6] argue that the notion of “cryptographic strength” captures the important facets of random sequences. To demonstrate cryptographic strength they follow this schema:

1. Upper bound the computational resources by Resources A.
2. Assume that Problem B cannot be solved within the limits of Resources A.
3. Produce a Pseudo-Random Bit Generator G
4. Argue that if an opponent sees the first m_0 bits generated by Pseudo-Random Bit Generator G and can utilize Resources A to predict the remaining bits with an accuracy rate of $1/2 + \epsilon(m)$ (where m is the size of the seed and ϵ is a fixed function satisfying $\lim_{m \rightarrow \infty} \epsilon(m) = 0$), then the opponent will be able to solve Problem B limited to Resources A by consulting the bit-guessing oracle, a contradiction.

Several cryptographically-strong pseudo-random bit generators have been proposed (BLUM-BLUM-SHUB[5], BLUM-MICALI[6],) and many applications have been discussed (ALEXI-CHOR-GOLDREICH-SCHNORR[3], GOLDREICH-GOLDWASSER-MICALI[9], GOLDWASSER-MICALI-TONG[10], VAZIRANI-VAZIRANI[20], YAO[22].) These generators are all inherently sequential, require polynomial time, and their cryptographic strength relies on some unproven cryptographic assumption.

Notation

When we say a class of circuit is *uniform*, we mean that it is constructible in logarithmic space by a deterministic Turing Machine.

NC (NC_U) is the class of languages accepted by (uniform, respectively) deterministic circuits with poly-log depth and polynomial size.

RNC (RNC_U) is the class of languages accepted by (uniform, respectively) randomized circuits with one-sided error, poly-log depth, polynomial size, and acceptance probability greater than $1/2$.

We give more precise definitions of these terms in section 4.

Our Result

We present a new cryptographically-strong pseudo-random bit generator which runs in NC_U but which is secure against attacks taking parallel poly-log time if the multiplicative inverse problem almost always is not in RNC . While we use the schema described above for demonstrating the cryptographic strength of our random number generator, because of the inherent parallel nature of our generator, the technical details of our proof are quite different from those of previous proofs for sequential pseudo-random number generators. In particular, we prove that if the bits output by our pseudo-random bit generator can be predicted in NC , then we can solve the multiplicative inverse problem in RNC almost always and this requires that we construct an interesting, nontrivial, parallel algorithm for that problem. (See section 3.)

About the Assumption

While our assumption has not been proved, it is quite interesting to observe that it is *testable* in the following sense: If a RNC algorithm takes more than poly-log time using our pseudo-random bits instead of truly random bits then we can observe this event by timing. Thus one of two scenarios is possible: either every application of our generator to a RNC algorithm yields a poly-log algorithm using only a small number of random bits, or some application of our generator is discovered to exceed its poly-log time bounds and we can immediately derive a NC algorithm for multiplicative inverse.

About the Measure of Randomness

VALIANT-SYKUM-BERKOWITZ-RACKOFF[19] show that an NC-machine can evaluate any straight-line program which computes a multivariate polynomial which has degree polynomial in the length of the program. Thus if our assumption is correct, our pseudo-random bit generator is secure against any statistical test which can be so formulated as a straight-line program. This includes most standard statistical tests for random number generators. (KNUTH[14])

Applications

Our method for parallel pseudo-random bit generation is actually very practical. It requires, for any $\epsilon > 0$, only $O(\log n(\log \log n))$ added depth and a factor of n^ϵ for a bounded fan-in circuit. Here is an example: KARP-WIGDERSON[12] gives a deterministic algorithm for the maximal independent set problem in $O((\log n)^4)$ time using $O(n^3/(\log n)^3)$ processors. They also give a uniform randomized algorithm for the same problem running in $O((\log n)^3)$ expected time with $O(n^2)$ processors using $O(n^2)$ random bits. Our results immediately yield an uniform algorithm with $O((\log n)^3)$ running time and $O(n^{2+\epsilon'})$ processors using only n^ϵ random bits, where $\epsilon, \epsilon' > 0$ can be set arbitrarily small.

Recently KARP-UPFAL-WIGDERSON[13] have shown that finding a maximum graph matching is in RNC_U , and ANDERSON[2] has shown that finding a maximal path is in RNC_U . Our results also immediately yield efficient randomized uniform algorithms for these problems, using only n^ϵ bits for any $\epsilon > 0$.

In further work REIF-TYGAR[16], we have applied results given in this paper to prove randomness properties of rational linear iterative maps modulo 1.

Implications

An interesting theoretical application of our result is that RNC_U is contained within the class of languages recognized by uniform deterministic circuits of unbounded fan-in with poly-log depth and 2^{n^ϵ} size for any $\epsilon > 0$. (ADLEMAN[1] proved RNC_U is contained in (non-uniform) NC, but the previous best construction for bounding RNC_U by deterministic uniform circuits of poly-log depth required $2^{n(n)}$ size.) This extends a result of YAO[22] for sequential polynomial time computations to poly-log time parallel computations.

2. Definitions and Results

Notation

We use the following notation throughout the paper:

N A positive composite integer such that each prime factor of N is greater than N^c for a fixed $c > 0$.

\mathbf{Z}_N^* The multiplicative group of positive integers less than and relatively prime to N . (Note that the fact that N has only large factors implies that a random positive integer less than N is an element of \mathbf{Z}_N^* with high probability.)

We will sometimes use $x \bmod N$ to indicate the residue of x modulo N .

Definitions

A **NC-machine** (COOK[8]) is a deterministic parallel algorithm which runs on $n^{O(1)}$ P-RAM processors in time $(\log n)^{O(1)}$ for input of size n . (Note that \mathbf{NC}_U is the class of languages accepted by NC-machines.)

A **RNC-machine** is a randomized parallel algorithm which runs on $n^{O(1)}$ P-RAM processors in time $(\log n)^{O(1)}$ for input of size n . (Note that \mathbf{RNC}_U is the class of languages accepted by RNC-machines.)

Given $s_0 \in \mathbf{Z}_N^*$, the *multiplicative inverse* of s_0 modulo N is the s_0^{-1} such that $s_0 s_0^{-1} = 1 \bmod N$.

For a fixed N , given an arbitrary $k \in \mathbf{Z}_N^*$, the multiplicative inverse problem is to find the multiplicative inverse of k modulo N . Note that the input size to the problem is $n = \lceil \log N \rceil$.

The problem of finding multiplicative inverses in poly-log depth has been studied extensively. (COOK[8], KANNAN-MILLER-RANDOLF[11], REIF[15], VON ZUR GATHEN[21].) Based on the lack of significant positive results obtained so far we conjecture:

Complexity Hypothesis

There exists an infinite sequence of numbers N_1, N_2, \dots constructable in \mathbf{NC}_U such that for each $n = 1, 2, \dots$ we have $n = \lceil \log N_n \rceil$ and that for almost all n , no **RNC-machine** exists

which can on arbitrary input from $Z_{N_n}^*$ solve the multiplicative inverse problem on any of those elements.

(Actually we could replace this complexity assumption with the weaker assumption that there exists a k such that for almost all n there exists an n' such that $n < n' < n^k$ and no RNC-machine can solve multiplicative inverse problem where the input can again range over arbitrary elements of $Z_{N_n}^*$. All the theorems in this paper would remain true under that weaker assumption.)

Definitions

A set S of bit sequences $\sigma = (b_1, \dots, b_J)$ of length $J = n^{O(1)}$ pseudo-random bits is **RNC-cryptographically strong** if no RNC-machine can, on a random input $b_1, \dots, b_i \in \sigma$ ($i < J, \sigma \in S$) predict any one bit b_{i+1}, \dots, b_J with expected success of $1/2 + 1/n^{O(1)}$. Informally, the bit sequences are **RNC-cryptographically strong** if no RNC-machine can predict untransmitted bits with an expected success rate significantly better than $1/2$.

Theorem

If the complexity hypothesis holds there exists a deterministic NC-machine \mathcal{G} which on an input seed of n bits outputs a **RNC-cryptographically strong** sequence of $J = n^{O(1)}$ pseudo-random bits. \mathcal{G} can be computed by a bounded fan-in uniform boolean circuit of depth $O((\log n)(\log \log n))$ and size $n^{O(1)}$.

This theorem is proved in section 3.

Definition

A **RNC-statistical test** is a **RNC-machine** which attempts to distinguish truly random bit sequences from pseudo-random bit sequences. A statistical test succeeds if it correctly distinguishes the pseudo-random bit sequences from truly random bit sequences with probability at least $1/n^{O(1)}$.

By a technique due to YAO[22] we can show that no RNC statistical test can succeed on **RNC-cryptographically strong** bit sequences. Hence:

Corollary 1

If the complexity hypothesis holds then no RNC-statistical test can succeed on our pseudo-random bit generator \mathcal{G} .

Corollary 2

If the N_n are constructable in depth $h(n)$, then given a randomized parallel algorithm \mathcal{A} (over a wide class of machine models such a parallel RAMS and fixed connection networks) with time bound $T(n)$ and processor bound $P(n)$, then \mathcal{A} can be simulated by a parallel algorithm with time bound $T(n) + h(n) + O((\log n)(\log \log n))$, processor bound $P(n)n^{\epsilon'}$, and only using n^{ϵ} truly random bits for any $\epsilon > 0$, where $\epsilon' = O(\epsilon)$.

$\text{CIRCUIT}_U(D(n), S(n))$ is the class of languages accepted by uniform deterministic circuits with unbounded fan-in, depth $D(n)$, and size $S(n)$. (See section 4 for a precise definition of these complexity classes.)

Corollary 3

If the complexity hypothesis holds then

$$\text{RNC}_U \subseteq \bigcup_{\epsilon > 0} \bigcap_{\epsilon' > 0} \text{CIRCUIT}_U((\log n)^{\epsilon}, 2^{n^{\epsilon'}})$$

This corollary is proved in section 4.

Corollary 4

There exists a cryptosystem where encryption and decryption can be done by a NC-machine on $n^{O(1)}$ bits given a secret shared key exactly n bits long (here n is a security parameter). If no RNC-machine can solve the multiplicative inverse problem then no RNC-machine can decrypt ciphertext exchanged in this cryptosystem.

We use the pseudo-random bits as a “one-time pad” — we take the sequential exclusive-or of the plaintext and the pseudo-random bits to produce the ciphertext and take the sequential exclusive-or of the ciphertext and the pseudo-random bits to obtain the plaintext again. Encryption and decryption both take parallel poly-log time but an opponent cannot decrypt the ciphertext with RNC-machine.

3. The Proof of the Main Theorem

Properties

We recall the following facts which we use implicitly (BEAME-COOKE-HOOVER[4], REIF[15], SHONHAGE-STRASSEN[18]):

- There exists a NC-machine for multiplication of two numbers in \mathbf{Z}_N^* .
- $2 \log p$ multiplications suffice to find the p^{th} power of a number in \mathbf{Z}_N^* .
- If $p < (\log N)^{O(1)}$, there exists a NC-machine for finding the p^{th} power of a number in \mathbf{Z}_N^* .

Fix $m = \lceil \log N \rceil$ throughout this section.

Let \mathcal{G} be the NC-machine which performs the following operations:

Input: random elements $s_0, k \in \mathbf{Z}_N^*$.

Output: b_1, \dots, b_J where $J = m^{O(1)}$.

Method: In parallel each processor P_i ($i = 1, \dots, J$) calculates $s_i = ks_0^i \bmod N$ and $b_{J-i+1} = B(s_i)$ where

$$B(x) = \begin{cases} 0 & \text{if } x \leq N/2 \\ 1 & \text{if } x > N/2 \end{cases}$$

Lemma

If there exists a RNC-machine which can determine the value of b_J with probability 1 (i.e., no error) on input b_1, \dots, b_{J-1} , then there exists a RNC-machine which can solve the multiplicative inverse problem for $\mathbf{Z}_{N_n}^*$.

Proof of Lemma

Suppose that MB (for “magic box”) is an oracle which can determine the value of b_J with probability 1. Then given $s_0 \in \mathbf{Z}_N^*$ we can find $s_0^{-1} \bmod N$. We can find this by running in parallel the following algorithm on each processor P_j for $(0 \leq j \leq m)$:

Set $k \leftarrow 2^j$. In parallel set $b_i \leftarrow B(ks_0^{J-i-1})$ for $1 \leq i \leq J-1$. Note that $b_J = B(2^j s_0^{-1})$. Feed the sequence (b_1, \dots, b_{J-1}) to MB to get b_J . Set the j^{th} most significant bit of δ to be

$B(2^j s_0^{-1})$. Define

$$\phi(\delta) = \left\lceil \frac{\delta N}{2^m} \right\rceil$$

Then $\phi(\delta) = s_0^{-1} \bmod N$. \square

Theorem

If there exists a RNC-machine which can determine the value of b_j with probability at least $1/2 + 1/m^{O(1)}$ on input b_1, \dots, b_{j-1} then there exists a RNC-machine \mathcal{M} which can solve the multiplicative inverse problem for $\mathbf{Z}_{N_n}^*$. \mathcal{M} can be computed by a bounded fan-in boolean circuit of depth $O((\log n)(\log \log n))$ and size $n^{O(1)}$.

Proof of Theorem

Assume that there exists a RNC-machine MB which can predict b_j with probability $1/2 + 2/m^c$. Let $H = 2(c+1) \lceil \log m \rceil$. Let δ and ϕ be as in the proof of the lemma.

Let $S = \{0, 1, \dots, 2^{H-1} - 1\}$. For each $0 \leq y < x \leq m$, we will create, by randomized methods, two functions $F_{x,y} : S \rightarrow \{0, 1\}^{x-y}$ and $G_{x,y} : S \rightarrow S$. Informally, values in S are guesses; $F_{x,y}$ is a rule for transforming a guess $j_x \in S$ into the x^{th} to y^{th} most significant bits of δ ; and $G_{x,y}$ is a rule for transforming the guess $j_x \in S$ into the guess $j_y \in S$.

If a RNC-machine could find δ for arbitrary s_0 , we could solve the multiplicative inverse problem. It will turn out that for some $j_m \in S$, that $F_{m,0}(j_m) = \delta$ with probability $1/2$. We can verify this occurrence simply by checking whether $s_0 \phi(\delta) = 1 \bmod N$. If we don't immediately find $s_0^{-1} \bmod N$, we simply form a new $F_{m,0}$ by randomized methods, and continue testing until we do find $s_0^{-1} \bmod N$.

Suppose we can determine j_x such that we know that $(2^x s_0^{-1} \bmod N)$ belongs to one of the two intervals

$$\left\{ \left\lceil \left\lfloor \frac{j_x N}{2^H} \right\rfloor \right\rceil, \left\lceil \frac{(j_x + 1)N}{2^H} \right\rceil \right\}, \left\{ \left\lceil \frac{(2^{H-1} + j_x)N}{2^H} \right\rceil, \left\lceil \frac{(2^{H-1} + j_x + 1)N}{2^H} \right\rceil \right\}$$

We can pick 2^H random values $\beta \in \mathbf{Z}_N^*$ and let v be MB's prediction for

$$B(2^x s_0^{-1} - \left\lfloor \frac{N j_x}{2^H} \right\rfloor + \beta \bmod N).$$

When β lies in the interval

$$\left[0, \left\lceil \frac{(2^{H-1} - 1)N}{2^H} \right\rceil \right],$$

mark a vote for v , when β lies in the interval

$$\left[\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{(2^H - 1)N}{2^H} \right\rfloor \right],$$

mark a vote the complement of v , and mark a *null* vote when β lies in other intervals. By assumption, MB predicts correctly with probability at least $1/2 + 2/m^c$.

We can assign a processor to calculate MB's prediction for each of the 2^H randomly chosen values of $\beta \in \mathbf{Z}_N^*$. This computation can be done in poly-log time for each β . The expected fraction of *null* votes is $2^{1-H} < 1/m^c$. Thus we have a bias of at least $2/m^c - 1/m^c = 1/m^c$ between 0 and 1 votes. Set $F_{x,x-1}(j_x)$ (our guess for $B(2^x s_0^{-1} \bmod N)$) to be a value which got the most votes. If our guess for $B(2^x s_0^{-1} \bmod N)$ is right, this immediately identifies which of the two intervals that $(2^x s_0^{-1} \bmod N)$ belongs to. 2^H tests are sufficient to make our guess correct with probability at least $1 - 1/2^m$. This result follows immediately from Chernoff bounds (CHERNOFF[7]); full details will appear in the complete paper. If our guess is right, that immediately determines the value of j_{x-1} ; that is, we can determine that $(2^{x-1} s_0^{-1} \bmod N)$ lies in one of the two intervals

$$\left\{ \left[\left\lceil \frac{j_{x-1}N}{2^H} \right\rceil, \left\lfloor \frac{(j_{x-1} + 1)N}{2^H} \right\rfloor \right], \left[\left\lceil \frac{(2^{H-1} + j_{x-1})N}{2^H} \right\rceil, \left\lfloor \frac{(2^{H-1} + j_{x-1} + 1)N}{2^H} \right\rfloor \right] \right\}$$

namely

$$j_{x-1} = G_{x,x-1}(j_x) = \lfloor j_x/2 \rfloor + 2^{H-2}(F_{x,x-1}(j_x))$$

We can calculate in parallel, for each $m \geq x \geq 1$, the functions $F_{x,x-1}$ and $G_{x,x-1}$, since the domain is finite and of polynomial size. If $x - y > 1$, then $F_{x,y}$ and $G_{x,y}$ can be recursively defined as

$$F_{x,y}(j_x) = F_{x,y}(G_{x,z}(j_x))2^{x-z} + F_{x,z}(j_x)$$

and

$$G_{x,y}(j_x) = G_{x,y}(G_{x,z}(j_x))$$

where $z = \lceil (x+y)/2 \rceil$. For each x, y pair ($0 \leq y < x \leq m$) and each $j_x \in S$ we repeatedly calculate the appropriate compositions of these functions for all j_x in the domain of the functions. Thus we can compute $F_{m,0}$ in $\lceil \log m \rceil$ stages.

Some guess j_m is correct. Suppose that for all $1 \leq i \leq m$, that (1) $G_{i,i-1}(j_i)$ is the correct value of j_{i-1} . Then (2) $F_{m,0}(j_m)$ would be the correct value of δ . For each i , the probability that (1) is true for a particular j_i is $(1 - 2^{-m})$, so the probability that (2) is true is $(1 - 2^{-m})^{m-1} > 1 - (m-1)2^{-m} > 1/2$.

For some $j_m \in S$, it will be true that $F_{m,0}(j_m) = \delta$ with probability $1/2$. We can try all possible j_m in parallel, and find out if we have a correct value by checking whether $\phi(F_{m,0}(j_m))s_0 = 1 \bmod N$. (Of course, it might happen that an incorrect guess for j_m might give a correct value for δ but this can only speed the calculation.) In the event that we do not get the correct value for δ mod N , we simply form new $F_{z,y}$ and $G_{z,y}$ functions and continue until we do get the correct value. \square

4. Randomized and Deterministic Parallel Complexity

Let \mathcal{C} be a list of circuits (C_1, C_2, \dots) of unbounded fan-in where C_n has n inputs and size $S(n)$. We consider \mathcal{C} to be *uniform* if there exists a $(\log S(n))$ space deterministic Turing machine which, given any n , outputs the circuit C_n . Let $\text{CIRCUIT}(D(n), S(n))$ be the class of all languages accepted by deterministic boolean circuits with unbounded fan-in, depth $D(n)$, and size $S(n)$. As usual we define

$$\text{NC} = \bigcup_{k_1 > 0, k_2 > 0} \text{CIRCUIT}((\log n)^{k_1}, n^{k_2})$$

We allow a *randomized boolean circuit* \mathbf{C} to have r special nodes each of which are assigned independent random bits chosen from $\{0, 1\}$ with equal probability. \mathbf{C} *accepts* an input $\omega \in \{0, 1\}^n$ if \mathbf{C} outputs 1 with probability $> 1/2$; otherwise \mathbf{C} *rejects* the input. For simplicity, we consider only one-sided error randomized circuits which never output a 1 on an input they have rejected. (The construction below can easily be extended to two-sided error randomized circuits which have an acceptance probability of at least $1/2 + 1/n^k$ for some $k > 1$.) Let $\text{RCIRCUIT}(D(n), S(n))$ be the class of languages accepted by randomized circuits with unbounded fan-in, depth $D(n)$, and size $S(n)$. We define

$$\text{RNC} = \bigcup_{k_1 > 0, k_2 > 0} \text{RCIRCUIT}((\log n)^{k_1}, n^{k_2})$$

We define CIRCUIT_U , NC_U , RCIRCUIT_U , and RNC_U analogously — restricting the circuits to be uniform.

Corollary 3

If the complexity hypothesis holds then

$$\text{RNC}_U \subseteq \bigcup_{c > 0} \bigcap_{c' > 0} \text{CIRCUIT}_U((\log n)^c, 2^{n^{c'}})$$

Proof

Let C be a (one-sided error) uniform randomized boolean circuit with n inputs, depth $D(n) = (\log n)^{k_1}$, and size $S(n) = n^{k_2}$. Fix any $\epsilon > 0$.

First suppose we had a source of $b = \lceil n^{\epsilon/2} \rceil$ truly random bits. Observe that C uses at most $S(n) = n^{k_2}$ random bits on each execution. Since $S(n) \leq b^{\epsilon'}$ where $\epsilon' = \lceil \epsilon/k_2 \rceil$ is constant, we can apply our parallel pseudo-random bit generator \mathcal{G} to produce $S(n)$ pseudo-random bits in $(\log n)^{O(1)}$ parallel time using $n^{O(1)}$ processors and using the b truly random bits as the seed. We can view the execution of C on the given input ω as a statistical test. By Corollary 2, given an input $\omega \in \{0, 1\}^n$, we need only execute C on ω for each of the 2^b possible pseudo-random bit sequences. We accept ω if C ever outputs 1.

Furthermore, we can avoid the use of a truly random seed by simply (1) enumerating all b -bit numbers in parallel; (2) executing the parallel pseudo-random bit generator using each of the b -bit numbers as a seed; and (3) executing C in parallel on ω on each of the resulting pseudo-random bit sequences. If C ever outputs 1 we accept ω . The resulting uniform circuit requires size $2^{b^2} \leq 2^{n^\epsilon}$ and depth $(\log n)^{O(1)} + O(D(n)) = (\log n)^{O(1)}$. \square

Note that if we require that our simulation circuit have bounded fan-in, then to simulate a circuit accepting a language in RNC_U , we require $n^{O(1)}$ (rather than $(\log n)^{O(1)}$ depth) and 2^{n^ϵ} size. This is an improvement over previous size bounds for RNC_U .

6. Acknowledgements

We would like to thank Michael Rabin for being an inspiration to us in the fields of randomized algorithms and cryptography.

We are indebted to Silvio Micali for his many helpful and insightful comments on this manuscript.

Also, thanks to Benny Chor, Shafi Goldwasser, Johan Hastad, Brian O'Toole, Charles Rackoff, Les Valiant, and Vijay Vazirani for their comments.

7. Bibliography

- [1] L. ADLEMAN *Two Theorems on Random Polynomial Time*, Proc. 19th IEEE Symposium on Foundations of Computer Science, Ann Arbor, MI, October 1978, pp. 75 – 83.

- [2] R. ANDERSON, *A Parallel Algorithm for the Maximal Path Problem*, Proc. 17th ACM Symposium on Theory of Computing, Providence, RI, May 1985, pp. 33 – 37.
- [3] W. ALEXI, B. CHOR, O. GOLDBREICH, AND C. SCHNORR, *RSA/Rabin Bits Are $1/2 + 1/\text{poly}(\log N)$ Secure*, Proc. 25th IEEE Symposium on Foundations of Computer Science, Singer Island, FL, October 1984, pp. 449 – 457.
- [4] P. BEAME, S. COOK, AND H. HOOVER, *Small Depth Circuits for Integer Products, Powers, and Division*, Proc. 25th IEEE Symposium on Foundations of Computer Science, Singer Island, FL, October 1984, pp. 1 – 6.
- [5] L. BLUM, M. BLUM, AND M. SHUB, *A Simple Secure Pseudo-Random Number Generator*, Proc. of CRYPTO-82, Santa Barbra, CA, September 1982, pp. 112 – 117.
- [6] M. BLUM AND S. MICALI, *How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits*, SIAM J. Comp., 13 (1984), pp. 850 – 864.
- [7] H. CHERNOFF, *A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations*, Ann. Math. Statist., 23 (1952), pp. 493 – 507.
- [8] S. COOK, *Towards a Complexity Theory of Synchronous Parallel Computation*, (Presented at) Inter. Symp. Logic. Alg. (1980).
- [9] O. GOLDBREICH, S. GOLDWASSER, AND S. MICALI, *How to Construct Random Functions*, Proc. 25th Symposium IEEE Symposium Foundations of Computer Science, Singer Island, FL, October 1984, pp. 464 – 479.
- [10] S. GOLDWASSER, S. MICALI, AND P. TONG, *Why and How to Establish a Private Code on a Public Network*, Proc. 23rd IEEE Symposium Foundations of Computer Science, Chicago, IL, October 1982, pp. 134 – 144.
- [11] R. KANNAN, G. MILLER, AND L. RUDOLF *Sublinear Parallel Algorithms for the Greatest Common Divisor of Two Integers*, Proc. 25th IEEE Symposium Foundations of Computer Science, Singer Island, FL, October 1984, pp. 7 – 11.
- [12] R. KARP AND A. WIGDERSON, *A Fast Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 16th ACM Symposium on Theory of Computation, Washington, DC, May 1984, pp. 266 – 272.

- [13] R. KARP, E. UPFAL, AND A. WIGDERSON, *Constructing a Perfect Graph Matching in RNC*, Proc. 17th ACM Symposium on Theory of Computing, Providence, RI, May 1985, pp. 22 – 32.
- [14] D. KNUTH, *The Art of Computer Programming, vol. 2: Seminumerical Algorithms, 2nd ed.*, Addison-Wesley, Reading, MA, 1981.
- [15] J. REIF, *Logarithmic Depth Circuits for Algebraic Functions*, Proc. 24th Symposium IEEE Foundations of Computer Science, Tuscon, AZ October 1983, pp. 138 – 145. Revised in Technical Report TR-84-18, Center for Research in Computing Technology, Harvard University. To appear in SIAM J. Comp.
- [16] J. REIF AND J. TYGAR, *The Complexity of Chaotic Iterative Maps*. To appear.
- [17] A. SHAMIR, *On the Generation of Cryptographically Strong Pseudo-Random Sequences*, ACM Trans. on Comp. Sys., 1, (1983), pp. 38-44.
- [18] A. SHONHAGE AND V. STRASSEN, *Schnelle Multiplication grosser Zahlen*, Computing, 7 (1974), pp. 281 – 292.
- [19] L. VALIANT, S. SYKUM, S. BERKOWITZ, AND C. RACKOFF, *Fast Parallel Computation of Polynomials Using Few Processors*, SIAM J. Comp., 12 (1983), pp. 641 – 644.
- [20] U. VAZIRANI AND V. VAZIRANI, *Trapdoor Pseudo-Random Number Generators with Applications to Protocol Design*, Proc. 24th IEEE Symposium Foundations of Computer Science, Tuscon, AZ, October 1983, pp. 23 – 30.
- [21] VON ZUR GATHEN, Private communication.
- [22] A. YAO, *Theory and Applications of Trapdoor Functions*, Proc. 23rd IEEE Symposium Foundations of Computer Science, Chicago, IL, October 1982, pp. 80 – 91.

How to Construct Pseudo-random Permutations from Pseudo-random Functions

Michael Luby

Charles Rackoff

Department of Computer Science

University of Toronto

Toronto, Canada M5S 1A4

ABSTRACT

Let F^n be the set of all functions from n bits to n bits. Let f^n specify for each key k of a given length a function $f_k^n \in F^n$. We say f^n is **pseudo-random** if the following two properties hold:

- (1) Given a key k and an input α of length n , the time to evaluate $f_k^n(\alpha)$ is polynomial in n .
- (2) If a random key k is chosen, f_k^n "looks like" a random function chosen from F^n to any algorithm which is allowed to evaluate f_k^n at polynomial in n input values.

Let P^{2n} be the set of permutations (1-1 onto functions) from $2n$ bits to $2n$ bits. Let p^{2n} specify for each key k of a given length a permutation $p_k^{2n} \in P^{2n}$. We present a simple method for describing p^{2n} in terms of f^n . The method has the property that if f^n is pseudo-random then p^{2n} is also pseudo-random. The method was inspired by a study of the security of the Data Encryption Standard. This result, together with the result of Goldreich, Goldwasser and Micali [GGM], implies that if there is a pseudo-random number generator then there is a pseudo-random invertible permutation generator. We also prove that if two permutation generators which are "slightly secure" are cryptographically composed, the result is more secure than either one alone.

The Bit Security of Modular Squaring given Partial Factorization of the Modulus

Benny Chor[†] Oded Goldreich[‡] Shafi Goldwasser^{*}

MIT, Laboratory for Computer Science

Cambridge, MA 01239

Abstract — It is known that given a composite integer $N = p_1 p_2$ (such that $p_1 \equiv p_2 \equiv 3 \pmod{4}$), and q a quadratic residue modulo N , guessing the least significant bit of a square root of q with any non-negligible advantage is as hard as factoring N .

In this paper we extend the above result to multi-prime numbers $N = p_1 p_2 \cdots p_l$ (such that $p_1 \equiv p_2 \equiv \cdots \equiv p_l \equiv 3 \pmod{4}$). We show that given N and q , a quadratic residue mod N , guessing the least significant bit of a square root of q is as hard as completely factoring N . Furthermore, the difficulty of guessing the least significant bit of the square root of q remains unchanged even when all but two of the prime factors of N , p_3, \dots, p_l , are known. The result is useful in designing multi-party cryptographic protocols.

1. Introduction

The problem of factoring large composite integers is perhaps the single most important computational problem in public key cryptography, as is evident from the large number of cryptosystems based on it (e.g. RSA [15], Rabin [13], Williams [18], Goldwasser-Micali [10]). The importance of the factoring problem motivated various research efforts. Among those are

- 1) Designing more efficient factorization algorithms.
- 2) Investigating the security of specific bits in the modular squaring function.
- 3) Investigating factorization algorithms given partial information on the factors [14].

Most of these works have concentrated on composite numbers N which are the product of two primes $p_1 p_2$.

In this paper we investigate the problem of bit security for the modular squaring function with respect to multi-prime composites $N = p_1 p_2 \cdots p_l$. The salient property of our work is that we investigate the bit security given partial factorization p_3, \dots, p_l of N (i.e. all but two

[†] Supported in part by an IBM Graduate Fellowship and a Bantrell Postdoctoral Fellowship.

[‡] Supported in part by a Weizmann Postdoctoral Fellowship. On leave from the Computer Sc. Dept., Technion

^{*} Supported in part by an IBM Faculty Development Award (1984) and NSF Grant DCR-8509905.

factors are known). We show that *the partial factorization does not help*. More specifically, any non-negligible advantage in guessing the least significant bit in the $x^2 \pmod{N}$ function is equivalent to factoring the remaining pair $p_1 p_2$ (and thus totally factor N). In other words, if it is infeasible to factor two prime composites, then it is infeasible to guess the least significant bit in the squaring modulo N function even if one has almost all of N 's factors.

Our work extends the results of Alexi, Chor, Goldreich and Schnorr [1], who considered the bit security of RSA and Rabin functions. These two functions are defined with respect to two-prime moduli $N = p \cdot q$. The RSA function is defined as raising to a power e and reducing modulo N (where e and $(p-1)(q-1)$ are relatively prime). Rabin's function is squaring modulo N . The RSA is 1-to-1, while Rabin's function is 4-to-1. This difference is crucial in trying to extend the [1] results to multi-prime moduli. Extending the RSA result to multi-prime moduli is easy, since the extended function is still 1-to-1. In the case of Rabin's function, squaring modulo an l -prime moduli is a 2^l -to-1 function, and dealing with it is more complicated. In this paper, we demonstrate how these complications can be resolved.

Our results have applications in the design of multi-party cryptographic protocols. In particular, it is useful in contexts where partial factorization, but not complete factorization, is released to a subset of the participants, while certain information must still be kept secret. Combining our result with techniques of probabilistic encryption [10,5], arbitrary information can be encoded so that it still remain totally secure, in such circumstances.

The remaining of this paper is organized as follows. In section 2 we introduce notations and terminology. In section 3 we review previous related results. In section 4 the main result is proved. In section 5 we mention two applications to the design of multi-party cryptographic protocols. We conclude by proposing an open problem.

2. Terminology

We begin this section by presenting some number theoretic terminology which will be used throughout the paper. We proceed by defining a specific class of composite integers which will constitute the domain of our investigation. We conclude this section by formally defining the notion of a "factoring bit".

2.1 Preliminaries

Definition 1: Let N be a natural number. \mathbf{Z}_N will denote the ring of integers modulo N , where addition and multiplication are done modulo N . The length of N will be denoted by n .

Definition 2: Let N be a natural number, and x an integer. $[x]_N$ will denote the remainder of x modulo N (notice that for all x , $0 \leq [x]_N < N$). $L_N(x)$ will denote the least significant bit of

$[x]_N$ in the ordinary binary expansion.

Definition 3: Let N be an integer. Then a is said to be a *quadratic residue modulo N* if there exist an integer x such that $x^2 \equiv a \pmod{N}$. Otherwise, a is said to be a *quadratic non-residue modulo N* . Let us denote by Q_N the set of quadratic residues modulo N .

Let $N = p_1 p_2 \cdots p_l$ be a product of l distinct odd primes. Note that a is a quadratic residue modulo N if and only if a is a quadratic residue modulo each of the p_i 's.

Definition 4: Let p be an odd prime number, and h an integer relatively prime to p . The *Legendre symbol* $\left(\frac{h}{p}\right)$ is defined to be 1 if h is a quadratic residue modulo p , and -1 otherwise. For $N = p_1 p_2 \cdots p_l$, a product of l distinct odd primes, and h relatively prime to N , the *Jacobi symbol* $\left(\frac{h}{N}\right)$ is defined to be $\prod_{i=1}^l \left(\frac{h}{p_i}\right)$.

Even though the definition of the Jacobi symbol uses the factorization of N , it is well known that $\left(\frac{h}{N}\right)$ be easily computed even if N 's factorization is not given. Another fact which is used in this paper is the multiplicativity of the Jacobi symbol, namely $\left(\frac{h \cdot h'}{N}\right) = \left(\frac{h}{N}\right) \cdot \left(\frac{h'}{N}\right)$. For further details on these properties and their proofs, see [12, ch. 3].

2.2 Blum Integers

When all the prime factors of $N = p_1 p_2 \cdots p_l$ are congruent to 3 (mod 4), the set of quadratic residues modulo N has an interesting property. Each quadratic residue has *exactly* one square root which is a quadratic residue itself. In other words, squaring modulo N is a permutation over Q_N . Blum was the first to point out the cryptographic significance of this fact [3]. Let $BI = \{N | N = p_1 \cdot p_2 \cdots p_l, p_i \equiv 3 \pmod{4}, 1 \leq i \leq l\}$, and call $N \in BI$ *Blum Integers*.

Definition 5: Let $N = p_1 p_2 \cdots p_l$ be in BI , and q be a quadratic residue modulo N . We denote by \sqrt{q} the square root of q which is a quadratic residue itself, namely $(\sqrt{q})^2 \equiv q$ and $\sqrt{q} \in Q_N$.

We restrict our attention to $N \in BI$, since for each quadratic residue $q \in Q_N$, \sqrt{q} and the least significant bit of \sqrt{q} are well defined.

2.3 Bit Security for Factoring

Following [6] and [11], we formally define the notion of bit security for factoring. For the definition, recall that n denotes the length of N .

Definition 6: Let \mathcal{O}_N be a probabilistic oracle which, given a quadratic residue q (modulo N), outputs a guess, $\mathcal{O}_N(q)$, for $L_N(\sqrt{q})$ (this guess might depend on the internal coin tosses of \mathcal{O}_N). Let $\epsilon(\cdot)$ be a function from integers into the interval $[0, \frac{1}{2}]$. We say that \mathcal{O}_N is a $\epsilon(n)$ -oracle if the probability that the oracle is correct, on an input q randomly selected from the set Q_N , is at least $\frac{1}{2} + \epsilon(n)$.

The probability space in the definition is that of all $q \in Q_N$ and all 0-1 sequences of internal

coin tosses, with uniform distribution. Notice that there is no requirements from the oracle if it is fed as input a number in \mathbb{Z}_N which is not a quadratic residue.

Definition 7: We say that *the least-significant bit of $\sqrt{\cdot}$ is $\epsilon(n)$ -secure* if there is a probabilistic polynomial time algorithm that on input $N, q \in Q_N$ and access to an arbitrary $\epsilon(n)$ -oracle for the least significant bit, O_N , computes \sqrt{q} .

Remarks: As is customary, we say that an algorithm is polynomial time if its running time is polynomial in its input length. In particular, the run time will be polynomial in n , the length (in binary) of the modulus N . In the last definition, the specific polynomial might depend on $\epsilon(\cdot)$. The same applies to the next definition.

Definition 8: We say that *the least-significant bit of $\sqrt{\cdot}$ is $\epsilon(n)$ -secure even if the factorization of N is partially known* if there is a probabilistic polynomial time algorithm that on input $N, q \in Q_N$, some (but not all) the prime factors of N and access to an arbitrary $\epsilon(n)$ -oracle for the least significant bit, O_N , computes \sqrt{q} .

We will subsequently replace $\epsilon(n)$ by ϵ for notational convenience. However, ϵ will still be a function of n .

3. Previous Results

In this section, we briefly review related previous results by Rabin [13], Blum, Blum and Shub [4], Alexi, Chor, Goldreich and Schnorr [1] and Vazirani and Vazirani [17].

3.1 The Equivalence of Factoring and Extracing Square Roots

Theorem 1 (Rabin): The following problems are probabilistic polynomial time equivalent

- 1) Factoring a composite integer N product of two primes.
- 2) Given N and $q \in Q_N$, finding a square root of q .

This Theorem easily extends to multi-prime integers.

3.2 Reducing Square Root Extraction to a Strange Oracle

Following a sequence of results in [11,2,16,9], Alexi, Chor, Goldreich and Schnorr [1] proved $1/\text{poly}(n)$ -security results for the least significant bit of a variant of the squaring modulo $N = p_1 p_2$ function. Their proof can be broken into two parts. First, a special type oracle, called (ϵ, q) -oracle is defined (see below). It is shown that factoring is in polynomial-time given access to an (ϵ, q) -oracle. Next, it was shown that an $(\epsilon/2, q)$ -oracle can be implemented using any ϵ -oracle for the least significant bit of a particular square root.

Definition 9: Let $N \in BI$ and $q \in Q_N$ be a quadratic residue. An (ϵ, q) -oracle is an oracle that

on input $s \in Z_N$ outputs $L_N(s \cdot \sqrt{q})$ with probability at least $\frac{1}{2} + \epsilon$. (Here the probability is taken over all possible choices of s and the internal coin tosses of the oracle with uniform probability distribution.)

The following Theorem is implicit in [1].

Theorem 2 (Alexi, Chor, Goldreich and Schnorr): There exists a probabilistic polynomial time algorithm that on input $N = p_1 p_2 \in BI$, $q \in Q_N$ and access to an arbitrary (ϵ, q) -oracle, finds \sqrt{q} .

The proof of Theorem 2 is almost identical to the proof in [1] of equivalence between inverting the RSA and guessing its least significant bit. While Theorem 2 deals with two prime composites, it extends to multi-prime composites. Combining the extended Theorems 1 and 2, we get

Corollary 1: There exists a probabilistic polynomial time algorithm that on input $N = p_1 p_2 \cdots p_l$, $q \in Q_N$ and access to an (ϵ, q) -oracle, completely factors N .

It is left to be shown that on input $N \in BI$, $q \in Q_N$ and access to an ϵ -oracle for $L_N(\sqrt{\cdot})$, an $(\epsilon/2, \cdot)$ -oracle can be implemented. This will be discussed in the next subsection.

3.3 Reducing the Strange Oracle to LSB Oracle when $N = p_1 p_2$

In this subsection we deal with implementing an (ϵ, q) -oracle, given access to an ϵ -oracle for $L_N(\sqrt{\cdot})$. The main difficulty lies in the fact that an (ϵ, q) -oracle must perform well when s ranges over Z_N while the ϵ -oracle is guaranteed to perform well only when its input ranges over Q_N .

The approach taken in resolving this difficulty is to map the queries to the (ϵ, q) -oracle into "queries" and "non-queries" to the ϵ -oracle. "Queries" are answered by invoking the ϵ -oracle, while "non-queries" are answered by flipping a coin. This requires the ability to distinguish "queries" from "non-queries". For $N = p_1 p_2$, two alternative implementations of this abstract approach were suggested.

The First Alternative

In [1], a slightly different predicate was considered (and shown to be equivalent to factoring). Instead of $L_N(\sqrt{\cdot})$ (the least significant bit of the square root which is a quadratic residue itself), they considered $B_N(\cdot)$, the least significant bit of the square root which has Jacobi Symbol 1 and is smaller than $N/2$. In the setting of [1] it is easy to test whether $[s \cdot \sqrt{q}]_N < N/2$ and whether the Jacobi Symbol $(\frac{s}{N})$ equals 1. Such s 's are mapped to "queries".

In the case of two-prime moduli each quadratic residue has a unique square root which satisfies the two conditions. However, in case the modulus has $l > 2$ factors, each quadratic residue has 2^{l-2} roots which satisfy the above two conditions. Thus, the solution of [1] to implementing the (ϵ, q) -oracle does not seem to extend to multi-prime moduli.

The Second Alternative

A different method of implementing the (ϵ, q) -oracle was suggested by Vazirani and Vazirani [17]. They observed that by Blum, Blum and Shub [4], the quadratic residuosity of s modulo a two-prime composite N can be determined by using an ϵ -oracle \mathcal{O}_N for the least significant bit. If $s \in Q_N$ then the ϵ -oracle for $L_N(s\sqrt{q})$ else a coin is flipped.

The advantage of this method is that the square root which is a quadratic residue itself is well defined also for multi-prime Blum integers. So there is hope of extending this method.

Let us recall how quadratic residuosity can be tested using an ϵ -oracle for $L_N(\sqrt{\cdot})$.

Theorem 3 (Blum, Blum and Shub): Let $N = p_1 p_2 \in BI$. There exist a probabilistic polynomial time algorithm that, on input $N, s \in Z_N$ and access to any ϵ -oracle for the least significant bit, \mathcal{O}_N , determines whether $s \in Q_N$.

Proof's sketch: If $(\frac{s}{N}) = -1$ then answer " $s \notin Q_N$ ". We are left with the case that $(\frac{s}{N}) = 1$. Consider the following experiment. Randomly select $r \in Q_N$ with uniform probability distribution (this is done by choosing an element in Z_N , with uniform probability, and squaring it). Let b be the oracle's answer on query $[(r \cdot s)^2]_N$. Clearly,

$$s \in Q_N \text{ implies } Pr(b = L_N(r \cdot s)) \geq \frac{1}{2} + \epsilon.$$

On the other hand, if $s \notin Q_N$ then $-\tau \cdot s \in Q_N$. As is always the case, $L_N(r \cdot s) = 1 - L_N(-\tau \cdot s)$ and thus

$$s \notin Q_N \text{ implies } Pr(b = L_N(r \cdot s)) \leq \frac{1}{2} - \epsilon.$$

So the two cases $s \in Q_N$ and $s \notin Q_N$ can be distinguished (with high probability) by sampling polynomially many r 's. ■

A crucial point in the proof is that for two-prime moduli $N = p_1 p_2$, $q \in Q_N$ has only two square roots with Jacobi Symbol $+1$. One of them is \sqrt{q} and the other is $-\sqrt{q}$. This is not the case when N has more than two prime factors. In fact, q has 2^{l-1} square roots which have Jacobi symbol $+1$. In the next section we show "a way around" this last problem.

4. The Main Result

In this section we implement an (ϵ, q) -oracle, given access to an ϵ -oracle to \mathcal{O}_N , where N is a multi-prime Blum integer. This, in turn, implies that an ϵ -oracle for the least significant bit, \mathcal{O}_N , enables the complete factorization of N .

Theorem 4: Let $N = M p_3 p_4 \cdots p_l$, $M = p_1 p_2$ and $N \in BI$, where the p_i 's are distinct odd primes. Then there is a probabilistic polynomial time algorithm that on input $N, q \in Q_N$, p_3, p_4, \dots, p_l and access to an arbitrary ϵ -oracle for the least significant bit, \mathcal{O}_N , implements an $(\epsilon/(2^l + 1), q)$ -oracle.

Proof: Let $Q'_N = \{e : \left(\frac{e}{p_1}\right) = \left(\frac{e}{p_2}\right) = -1 \text{ and } \left(\frac{e}{p_i}\right) = 1 \text{ for every } 3 \leq i \leq t\}$.

Given $q \in Q_N$, and access to the ϵ -oracle for the least significant bit, \mathcal{O}_N , we implement an $(2^{-l} \cdot \epsilon, q)$ -oracle as follows. On query $s \in Z_N$, we first compute the Jacobi Symbol $\left(\frac{s}{M}\right)$ and the Legendre Symbols $\left(\frac{s}{p_2}\right), \left(\frac{s}{p_4}\right), \dots, \left(\frac{s}{p_l}\right)$. If either of the above equals -1 then $s \notin Q_N \cup Q'_N$, and we return the outcome of an unbiased coin flip. It remains to deal with $s \in Q_N \cup Q'_N$. We consider two cases:

- Case I: The oracle \mathcal{O}_N answers to $L_N(s\sqrt{q})$ are considerably worse for $s \in Q'_N$, compared to $s \in Q_N$. In this case we first use \mathcal{O}_N to test whether $s \in Q_N$. Our answer to $L_N(s\sqrt{q})$ is $\mathcal{O}_N(s^2 \cdot q)$ if $s \in Q_N$, and a flip of a coin if $s \in Q'_N$.
- Case II: The oracle \mathcal{O}_N answers to $L_N(s\sqrt{q})$ are not considerably worse for $s \in Q'_N$, compared to $s \in Q_N$. In this case, we answer to $L_N(s\sqrt{q})$ by $\mathcal{O}_N(s^2 \cdot q)$. Intuitively, it does not matter here whether $s \in Q_N$ or $s \in Q'_N$.

To treat the above cases formally, we define the success probabilities of \mathcal{O}_N on query $[r^2]_N$ where $r \in Q_N$ (correspondingly $r \in Q'_N$) is randomly chosen. (The probabilities are taken over \mathcal{O}_N 's internal coin tosses.) Let

$$f = \Pr(\mathcal{O}_N(r^2) = L_N(r)) \quad \text{where } r \text{ is randomly chosen in } Q_N$$

$$f' = \Pr(\mathcal{O}_N(r^2) = L_N(r)) \quad \text{where } r \text{ is randomly chosen in } Q'_N.$$

By \mathcal{O}_N 's definition, $f \geq \frac{1}{2} + \epsilon$, but no a-priori bounds on f' are known.

With overwhelmingly high probability (say $1 - 2^{-n}$), both f and f' can be approximated with good accuracy (say $\epsilon/8$) by the following polynomial time Monte Carlo experiments: To approximate f , randomly select many independent $r \in Q_N$ with uniform probability distribution. (A random $r \in Q_N$ is selected by picking an element of Z_N at random and squaring it modulo N .) Compare \mathcal{O}_N 's answer on $[r^2]_N$ with the known $L_N(r)$. To approximate f' , randomly select many independent $r \in Q'_N$ with uniform probability distribution, and compare \mathcal{O}_N 's answer on $[r^2]_N$ with the known $L_N(r)$. A random $r \in Q'_N$ is selected by picking $r' \in Q_M$ and $r'' \in Q_{p_3 p_4 \dots p_l}$, at random, setting $r \equiv -r' \pmod{M}$ and $r \equiv r'' \pmod{p_3 p_4 \dots p_l}$, and computing r by the Chinese Remainder Theorem.

Let us denote the above approximations by \tilde{f} and \tilde{f}' respectively (i.e. $|f - \tilde{f}| < \epsilon/8$ and $|f' - \tilde{f}'| < \epsilon/8$ with overwhelming high probability). We now consider two cases

Case I: $\tilde{f}' < \tilde{f} - \epsilon/2$.

In this case we will use \mathcal{O}_N to test whether $s \in Q_N$. To do that, randomly select $r \in Q_N$ with uniform probability distribution. Let b be the oracle's answer on query $[(r \cdot s)^2]_N$. If $s \in Q_N$ then

$\Pr(b = L_N(\tau \cdot s)) = f$, while if $s \in Q'_N$ then $\Pr(b = L_N(\tau \cdot s)) = f'$. Since $|f - f'| > \epsilon/4$ (with overwhelming probability), the two cases can be distinguished by a Monte-Carlo experiment.

If we have decided that $s \in Q_N$ then we query the oracle on s^2q and return whatever it has answered (i.e. we return $O_N(s^2q)$). Otherwise, we flip an unbiased coin and return its outcome.

Case II: $\tilde{f}' \geq \tilde{f} - \epsilon/2$.

In this case we will not try to test whether $s \in Q_N$ or $s \in Q'_N$, but rather query O_N on s^2q and return $O_N(s^2q)$. Here $f' \geq \frac{1}{2}$ with overwhelming probability.

Probability Analysis

We now analyze the probability that the answer to $L_N(s\sqrt{q})$ produced by the above procedure is correct. The probability space is that of all choices of $s \in Z_N$ and all internal coin tosses with uniform distribution.

The event $s \notin Q_N \cup Q'_N$ occurs with probability $1 - 2 \cdot 2^{-l}$ and is always detected. In this case the above procedure is correct with probability exactly one half.

The event $s \in Q_N \cup Q'_N$ occurs with probability 2^{-l+1} . In Case I, the answer is correct with probability $\frac{1}{2}(\frac{1}{2} + f) \geq \frac{1}{2} + \frac{\epsilon}{2}$ (up to the overwhelmingly small error term of the approximations). In Case II, the answer is correct with probability $\frac{1}{2}(f + f') \geq \frac{1}{2} + \frac{\epsilon}{2}$ (with the same qualification). The overall probability that our procedure is correct is therefore bounded below by

$$\frac{1}{2} + \frac{1}{2^l} \cdot \epsilon - 2^{-n}.$$

Thus, we have implemented an $(\epsilon/(2^l + 1), q)$ -oracle ■

The proof of Theorem 4 shows how to implement an $(\epsilon/2^l, q)$ -oracle given an ϵ -oracle for the least significant bit $L_N(\cdot)$, where N has l prime factors. Thus, when $l = O(\log n)$ the advantage of the new oracle is polynomially (in n) related to the advantage of the original one. Combining Corollary 1 and Theorem 4, we get

Corollary 2: Let $N, M \in BI$ such that M divides N . Suppose that M has two prime factors and N has $l = O(\log n)$ distinct prime factors, where n is the length of N . Then the following two tasks (1) and (2) are computational equivalent, and both are polynomial-time reducible to (3).

- 1) Factoring M .
- 2) Given M, p_3, p_4, \dots, p_l (a partial factorization of $N = Mp_3p_4 \cdots p_l$) and $q \in Q_N$, guess $L_N(\sqrt{q})$ with success probability exceeding $\frac{1}{2} + \frac{1}{\text{poly}(n)}$.
- 3) Let $1 \leq k < l, N_1, N_2, \dots, N_k$ such that $N = N_1N_2 \cdots N_k$ and M divides N_1 . Given N_1, N_2, \dots, N_k and $q \in Q_N$, guess $L_N(\sqrt{q})$ with success probability exceeding $\frac{1}{2} + \frac{1}{\text{poly}(n)}$.

5. Applications to Protocols Design

Chor, Goldwasser, Micali, and Awerbuch [7] suggested to use a composite number N product of $l = 2^t + 1$ primes in order to "verifiably share" a secret bit among many players, t of which can be untrusty. They suggested two implementations of this scheme: One is based on the RSA, while the other is based on modular squaring. The security of the second implementation relies on the result of this paper. A brief description of the scheme follows.

The secret is the least significant bit of \sqrt{q} , where $q \in Q_N$ is a quadratic residue modulo N . After establishing the secret, the dealer distributes "pieces" of it to every participant (one piece per participant). A random split of N corresponds to one piece of the secret bit. Since N has $2^t + 1$ prime factors, it cannot be totally factored with only t pieces. By our result, it is infeasible for t participants to guess the secret $L_N(\sqrt{q})$ with any non-negligible advantage. On the other hand, with overwhelmingly high probability, $3t$ pieces yield the complete factorization of N and allow the recovery of the secret bit.

6. An Open Problem

A crucial condition for proof of Corollary 2, is that the number of prime factors is logarithmic in the length of the modulus. The reason being that the inverting algorithm needs answers for random elements in Z_N , while the ϵ -oracle for least significant bit answers only on $q \in Q_N$. Thus, only a 2^{-l} fraction of the queries are answered, where l is the number of primes in N . Getting around this difficulty will require either a different inverting algorithm or a better analysis of what happens when the oracle is asked on $q \in Z_N - Q_N$.

References

- [1] Alexi, W., B. Chor, O. Goldreich, and C.P. Schnorr, "RSA and Rabin Functions: Certain Bits are As Hard As The Whole", to appear in *SIAM Jour. on Computing*. Extended abstract in *Proc. of 25th FOCS*, 1984, pp. 449-457.
- [2] Ben-Or, M., B. Chor, and A. Shamir, "On the Cryptographic Security of Single RSA Bits", *15th ACM Symp. on Theory of Computation*, April 1983, pp. 421-430.
- [3] Blum, M., "Coin Flipping by Telephone", *IEEE Spring COMCON*, 1982.
- [4] Blum, L., M. Blum, and M. Shub, "Comparison of Two Pseudo-Random Number Generators", *Advances in Cryptology: Proceedings of Crypto82*, Chaum, D., et al. eds., Plenum Press, 1983, pp. 61-79.
- [5] Blum, M., and S. Goldwasser, "An Efficient Probabilistic PKCS as Secure as Factoring", *Advances in Cryptography: Proceedings of Crypto 84*, Springer Verlag, Lecture Notes in

- Computer Science (196), 1985, pp. 289-299.
- [6] Blum, M., and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM Jour. on Computing*, Vol. 13, No. 4, November 1984, pp. 850-864.
 - [7] Chor, B., S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults", *Proc. of 26th FOCS*, 1985, pp. 383-395.
 - [8] Diffie, W., and M.E. Hellman, "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.
 - [9] Goldreich, O., "On the Number of Close-and-Equal Pairs of Bits in a String (with Implications on the Security of RSA's L.s.b.)", MIT/LCS/TM-256, March 1984.
 - [10] Goldwasser, S., and S. Micali, "Probabilistic Encryption", *Jour. of Computer and System Science*, Vol. 28, No. 2, 1984, pp. 270-299.
 - [11] Goldwasser, S., S. Micali, and P. Tong, "Why and How to Establish a Private Code on a Public Network", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, November 1982, pp. 134-144.
 - [12] Niven, I., and H.S. Zuckerman, *An Introduction to the Theory of Numbers*, John Wiley & Sons Inc., (1980).
 - [13] Rabin, M.O., "Digital Signatures and Public Key Functions as Intractable as Factorization", MIT/LCS/TR-212, 1979.
 - [14] Rivest, R.L., and A. Shamir, "An Efficient Factoring Algorithm Based on Partial Information", presented in Eurocrypt85, Linz, Austria, April 1985.
 - [15] Rivest, R.L., A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signature and Public Key Cryptosystems", *Comm. of the ACM*, Vol.21, February 1978, pp. 120-126.
 - [16] Vazirani, U.V., and V.V. Vazirani, "RSA Bits are $.732 + \epsilon$ Secure", *Advances in Cryptology: Proceedings of Crypto83*, Chaum, D. ed, Plenum Press, 1984, pp. 369-375.
 - [17] Vazirani, U.V., and V.V. Vazirani, "Efficient and Secure Pseudo-Random Number Generation", *Proc. of 25th FOCS*, 1984, pp. 458-463.
 - [18] Williams, H.C., "A Modification of the RSA Public-Key Encryption Procedure", *IEEE Trans. Info. Th.*, IT-26 (1980), pp. 726-729.

SOME CRYPTOGRAPHIC ASPECTS OF WOMCODES

Philippe Godlewski and Gérard D. Cohen

ENST, Département SYC, 46 rue Barrault, 75013 PARIS, France.

Abstract

We consider the following cryptographic and coding questions in relation with the use of "write-once" memories (or woms)

-How to prevent anyone from reusing the wom (immutable codes).

-How to fix the written information in the wom after a given number of generations (locking codes).

-How to encode a "credit" in a way that guarantees the user t generations or "purchases" in any possible way and makes it impossible to cheat : i.e. writing on the wom necessarily increases the spent amount of money. The coding will be called "incremental locked".

These questions were only raised in [5], where the accent was put on the generation of womcodes possessing an "easy reading-reserved writing" property.

1. Definitions and notations

Let us suppose we have a storage medium, called wom ([1]), consisting of n binary positions or wits, initially containing a "0". At some step, a wit can be irreversibly overwritten with a "1" (e.g. by some laser beam in digital optical disks, or burning microscopic fuses in PROMS).

For two binary n -tuples x and y , we say that x covers y , and write $y \prec x$ if $\text{supp}(y) \subset \text{supp}(x)$, where for a binary n -tuple $z = (z_1, z_2, \dots, z_n)$, $\text{supp}(z) = \{i ; z_i = 1\}$ is the support of z . Then $|z| = |\text{supp}(z)|$ is the Hamming weight of z . The binary complement of z is denoted by \bar{z} .

The first problem we address is the following : how to construct codes with maximal rate (or cardinality) and forwarding impossible updating?

2. Immutable codes

Let F be the binary field. A subset C of F^n is called **immutable** (see [6]) if, for any a and b in C , $a < b$ never holds. Clearly, if such a code is used to write on a worn, no updating is possible (updating a into b would imply $a < b$). The characterization of maximal immutable codes is a well known combinatorial problem, solved by Sperner [2].

Proposition. The set S of all the n -tuples of weight $\lfloor n/2 \rfloor$ is a maximal immutable code (called **Sperner code**). The solution is unique for odd n . For even n , there is another solution \bar{S} , where $\bar{S} = \{ \bar{s} \in F^n ; s \in S \}$.

The rate of these codes, $R = (1/n) \cdot \log(|S|)$, is approximately

$$R \approx 1 - (1/2n) \log(n).$$

These Sperner codes are however not very easy to encode (see e.g. [7]). One way to overcome this is to impose linearity. This will be very suboptimal, as we now show. Let us say that a linear $[n, k]$ code C is **intersecting** if any two non-zero codewords have intersecting supports, then one has :

Proposition. A linear code C is intersecting iff $C \setminus \{0\}$ is immutable.

Proof. $C \setminus \{0\}$ is not immutable iff there exist two distinct nonzero elements in C , say a and b , with $a < b$. Then $a + b$ is in C and has disjoint support with a , hence C is not intersecting. \square

Intersecting codes are studied in, e.g. [3], and have low rate, namely :

Proposition. For n large enough, intersecting $[n, k]$ codes have rate $R < 0.283 \cdot n$.

We now propose a slightly suboptimal solution, first introduced in [7], with a very simple encoding scheme. Let us denote by $2(i)$ the writing of the integer i in base 2, and by $|2(i)|$ the weight of such a writing. Define the coding of $i < 2^k$ by

$$i \longrightarrow c(i) = (2(i), \overline{2(|2(i)|)}) \quad (1)$$

where the two parts of $c(i)$ are written using k and $\lceil \log(k) \rceil$ bits respectively. For example, if $k=7$, $i=98$, then $2(i) = 1100010$, $12(i)=3$, $2(12(i)) = 011$ and $c(98) = (1100010 \ 100)$.

In fact, this encoding is **systematic**, i.e. the information written on the wom is contained in k fixed positions, say the first ones. Clearly, one has :

Proposition. The encoding scheme described in (1) gives immutable codes with rate $R \approx 1 - (1/n) \log(n)$.

Proposition. The encoding scheme of (1) is optimal, i.e. yields the largest possible rate for a systematic immutable code.

Proof. Let C be systematic with k information bits. Consider the chain (for inclusion) of k -tuples $(000\dots 0)$, $(100\dots 0)$, $(110\dots 0)$, ..., $(111\dots 1)$. For C to be immutable, these $k+1$ vectors must be appended different suffixes of size $n-k$. Hence $2^{n-k} \geq k+1$. \square

We thank D. Coopersmith for suggesting this proof.

3. Locking codes

The problem of **locking**, i.e. of fixing the written information in the wom after a given number of generations, is closely related to the previous one. The only difference is that one now has the possibility of choosing when the written information should become immutable, which is a slightly stronger assumption. Among the techniques described in paragraph 2, the coding scheme (1) allows locking : to that end, take a wom of $k+\lceil \log k \rceil$ wits,

- use m wits for the updatings,
- to lock the wom when v is written, write $2(|v|)$ on the remaining wits.

4. Incremental locked codes

The following problem is introduced in [4] : write successively t messages v_1, v_2, \dots, v_t on a wom, such that

$$0 \leq v_1 \leq v_2 \leq \dots \leq v_t \leq v-1. \quad (2)$$

Such a code is called **incremental (IW)**.

We consider the problem where any writing on the wom can only increase the value of the written message. Such a code will be called a **incremental locked womcode (ILW)** and can be used to eliminate cheating possibilities on credit cards. This assumption is stronger than the previous one : now (2) is a necessary and sufficient condition on a set of t messages for its writing to be possible, whereas it was only sufficient in the case of IW.

We shall study in the following an easy way to construct a ILW : the **knapsack (or coins) scheme**. Each wit represents a coin with value a_i . Thus the spent amount of money corresponds to the sum of "marked" coins

$$v_j = \sum_{i \in I} a_i$$

where I is the set of written wits. We call **incremental K womcodes (IKW)** the corresponding codes. Clearly we have

$$w_{ik} \geq w_{il} \geq w_i$$

where w_{ik}, w_{il}, w_i are the minimal lengths of a IKW, ILW, IW, respectively.

We consider the directed graph (treillis) representing all the possible transitions in the WOM. A vertex is identified with a binary n -tuple, and there is an edge from x to y iff $y > x$ and $|y - x| = 1$. To every y is associated a message $\alpha(y) \in \mathbb{Z} \cup \{\omega\}$ by means of the interpreting function α : $\alpha(\omega)$ means that the state y is not used (achievable as a coding state) in the coding process. The incremental code is locked iff for achievable x and y

$$y > x \implies \alpha(y) \geq \alpha(x).$$

For every set

$$V = (v_1, v_2, \dots, v_t), \text{ with } v_1 \leq v_2 \leq \dots \leq v_t$$

of t messages to be written, we consider the "history" of writings

$$Y = (y^{(1)}, y^{(2)}, \dots, y^{(t)}) \text{ where } y^{(i)} \in \mathbb{Z}^n, y^{(1)} < y^{(2)} < \dots < y^{(t)}$$

$$\text{and } \alpha(y^{(i)}) = v_i.$$

Let H be the set of all possible Y . The number of possible V must be less than the number of possible Y . Thus we obtain :

Proposition. The parameters of a $\langle v \rangle^t / n$ IW must satisfy

$$\binom{v+t}{t} \leq (t+1)^n.$$

We now define for $y \in P^n$:

$$\theta(x) = \inf \{ i \mid x = y^{(i)} \text{ for some } Y = (y^{(1)}, \dots, y^{(i)}, \dots), Y \in H \}.$$

Proposition. If y is a state in the WOM such that $\theta(y) = j$, then $n - |y| \geq w(\langle v - \alpha(y) \rangle^{t-j}) + j$,

where λ stands for i, il, ik in the case of a IW, IL, IKW respectively.

Indeed, at state j , there are at least $t-j$ generations to write on $n - |y|$ wits.

Using this Proposition we can begin to fill up a table of the w^λ for small v and t . We start from the first line $w(\langle v \rangle^1) = \lceil \log_2(v) \rceil$. The noticeable points are

$$w^{ik}(\langle 9 \rangle^3) = 6 > 5 = w^{il}(\langle 9 \rangle^3) \quad \text{and} \quad w^{il}(\langle 9 \rangle^2) = 5 > 4 = w^i(\langle 9 \rangle^2).$$

$v \backslash t =$	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	2	2	3	3	3	3	4	4	4	4
2			2	3	3	4	4	4	4,5,5	5	5	5
3				3	4	4	5	5	5,5,6		6	6,6,7
4					4	5	5	6	6			
5						5	6	6	7			

Table : values of $w^i(\langle v \rangle^t)$, $w^{il}(\langle v \rangle^t)$, $w^{ik}(\langle v \rangle^t)$ for small v and t .

5. Construction of incremental K womcodes (IKW)

As we said before an incremental K womcode is based on a set of coins $P = \{\dots, i, \dots, j, \dots\}$, where i is a coin with value i and $|P| = n$. The set P is hereafter referred to as a **purse**. The coding algorithm obeys the following rule: "use first the heaviest remaining coin compatible with the purchase". We shall say that a $\langle (s+1)^t / n \text{ IKW} \rangle$ realizes (s, t) . Let us introduce some notations:

$n_j(P)$ is the number of coins in P with value j ;

$$\Sigma_i(P) = \sum_{j=1}^i j n_j, \quad \Sigma(P) = \Sigma_{\infty}(P);$$

P/i is the set of coins in P with value at most i ;

$$\text{then } |P/i| = \sum_{j=1}^i n_j(P) \text{ and } \Sigma_i(P) = \Sigma(P/i);$$

$Q_i[k]$ or Q_i : a purse with only k coins of value i (then $k = |Q_i| = n(Q_i)$);

$D = (d_1, d_2, \dots, d_t)$ a t -tuple of purchases; $\Sigma(D) = \sum_j d_j$.

In the following, P denotes a purse realizing (s, t) , and $m = \lfloor s/t \rfloor + 1$.

Proposition K1. For every integers $\mu \leq m, r$,

$$P^{(r)} = P \cup \bigcup_{\mu} Q[r] \text{ realizes } (s+r\mu, t).$$

Proof. By induction on r . Suppose it is true up to k i.e. $P^{(k)} = P \cup \bigcup_{\mu} Q[k]$

realizes $(s+k\mu, t)$. Let D be a t -tuple to be spent using $P^{(k+1)}$, let j_0 be the first j such that $d_j \geq \mu$ (if no such j_0 exists $\Sigma(D) \leq (\mu-1)t \leq s$ and we are done). Set

$$D' = (d'_j) = (d_1, d_2, \dots, d_{j_0} - \mu, \dots, d_t).$$

From our "heavy coin first" algorithm, realizing D with $P^{(k+1)}$ amounts to realizing D' with $P^{(k)}$, hence is possible since $\Sigma(D) \leq s+k\mu$. \square

Proposition K2. The purse P_i defined recursively by

$$P_1 = Q_1[t],$$

$$P_2 = P_1 \cup Q_2[n_2] \text{ where } n_2 \text{ is the smallest integer such that } \Sigma(P_2) \geq 2t,$$

...

$$P_i = P_{i-1} \cup Q_i[n_i] \text{ where } n_i \text{ is the smallest integer s.t. } \Sigma(P_i) \geq it,$$

realizes every t -tuple of purchases $D = (d_1, d_2, \dots, d_t)$ with $\Sigma(D) \leq \Sigma(P_i)$.

Proof. By induction. For any fixed j , $0 < j \leq i-1$, step $P_j \rightarrow P_{j+1}$ is achieved by applying Proposition K1 with $\mu=j+1$, $r=n_{j+1}$, $s=jt$ and therefore $m=j+1$. \square

Remark. The construction in Proposition K2 also works without assuming the n_j minimal. By stopping at some level k , we obtain purses P for which the following also holds

$$\Sigma(P/j) \geq jt, \forall j \text{ s.t. } 1 \leq j \leq k$$

or equivalently

$$\Sigma(P/j) \geq jt, \forall j \text{ s.t. } jt \leq \Sigma(P) \quad (*)$$

But (*) is at the same time a necessary condition for a purse P to realize $(\Sigma(P), t)$ because every t -tuple D with $\Sigma(D) \leq \Sigma(P)$ and $\text{Max}_k d_k \leq j$ must be realized with P/j . This shows :

Corollary. For given s and t , a necessary and sufficient condition for a purse P with $\Sigma(P/m) \geq s$, $m = \lfloor s/t \rfloor + 1$, to realize (s, t) is that the $m-1$ following t -tuples of purchases be realizable :

(j, j, \dots, j) for $1 \leq j < m$.

Optimality of the proposed construction

Now we want to prove that the purse defined by Proposition K2 is optimal in the class of IKW. For fixed t , a purse P is said **saturated** if P realizes $(\Sigma(P), t)$. We first show that we can restrict ourselves to saturated purses. As before, P denotes a purse realizing (s, t) , with $m = \lfloor s/t \rfloor + 1$.

Proposition. For any P realizing (s, t) , there exists a saturated P^0 such that $\Sigma(P^0) = s$ and $|P^0| \leq |P|$.

Proof. We first show that P/m realizes (s, t) : Consider $D = (d_i)$, $\Sigma(D) = s$ and $d_i \in \{m-1, m\}$. Such a set of purchases uses coins with value at most m , hence $\Sigma(P/m) \geq s$. Then apply Corollary, which shows that P/j realizes $(\Sigma(P/j), t)$ if $1 \leq j \leq m$.

Define m' by

$$\Sigma(P/(m'-1)) < s \leq \Sigma(P/m').$$

It is clear that $m' \leq m$. The purse $P' = Q[k] \cup (P/(m'-1))$ realizes $(\Sigma(P'), t)$ by proposition K1. Choose k s.t. $\Sigma(P') \leq s < \Sigma(P') + m'$. If the left-hand side inequality is achieved then $P^0 = P'$ is a desired purse. If not, consider $P^0 = P' \cup \{j\}$, $j = s - \Sigma(P')$, then P^0 realizes (s, t) , again by proposition K1, and $\Sigma(P^0) = s$. After straightforward counting, we get

$$|P^0| = |P/m'| - \lfloor (\Sigma(P/m') - s)/m' \rfloor \leq |P/m'| \leq |P|$$

We have transformed P into a saturated P^0 with fewer coins. \square

Let now $f(s, t)$ be the minimum number of coins for a purse realizing (s, t) : $f(s, t) = w_{ik}^t(\langle s+1 \rangle^t)$. Then we have:

Proposition. The purse P_i defined by Proposition K2 is optimal. That is, $f(\Sigma(P_i), t) = |P_i|$.

Proof. By induction on i . Suppose it is true up to $i-1$. We first recall that P_i is obtained from P_{i-1} by possibly adding coins with value i . Then setting $s_j = \Sigma(P_j)$, $s = s_{i-1}$ and $s' = s_i$, we have $s' - s = ki$ for some integer k . Let P be an optimal saturated purse realizing (s', t) ; therefore $P = P/i$ (see previous proof). From P we can construct, as before, a saturated P^0 realizing (s, t) by suppressing heaviest coins (with value at most i) and possibly adding a "cheaper" extra one.

$$|P^0| \leq |P| - \lfloor (s' - s)/i \rfloor.$$

Now if $|P| = f(s', t) < |P_{i-1}|$, then $f(s, t) < |P_{i-1}|$ and we get a contradiction. \square

6. Asymptotical results

For womcodes, the asymptotical behavior is studied in [1]. Focusing on the case when t is fixed and v goes to infinity, one has

$$w(\langle v \rangle^t) \approx f(t) \log_2(v),$$

with $f(2) \approx 1.29$ and $f(t) \approx t/\log_2(t)$ for t large.

Clearly, an incremental womcode realizing $(v+1, t)$ is also a $\langle (v+1)/t \rangle^t$ womcode. Hence, for fixed t

$$w(\langle v \rangle^t) \geq w(\langle (v+1)/t \rangle^t) \approx f(t) \log_2((v+1)/t) \approx w(\langle v \rangle^t).$$

That is, $w^i \approx w$ (cf. [4]).

From the previous section, we know that recursive purses yield incremental K womcodes with

$$(i+1)t > E(P_i) \geq it$$

and maximum coin of value $(i+1)$.

For fixed t and i going to infinity, the average increase of $E(P_i)$, $E(E(P_{i+1}) - E(P_i))$ is equal to t , or

$$E(|P_{i+1} - P_i|) = 1.$$

In others words, the purse P_i realizing (s_i, t) has j coins, with

$$j \approx \sum_{k=1}^i t/k \approx t \ln(i) \approx t \ln(s_i/t).$$

Finally, since these codes are optimal

$$w^{ik} \approx t (\ln(v) + O(1)).$$

The asymptotical behavior of w^{il} is still unknown. It would be interesting to estimate

$$R = \limsup w^{il} / w^{ik},$$

for fixed t and v going to infinity, and to prove that $R < 1$.

Let us summarize what we know about w .

	t large	t=2	t=3
no coding	$w_0 = t \log_2 v$	$2 \log_2 v$	$3 \log_2 v$
incremental K womcodes	$w \approx t \log_e v$	$1.38 \log_2 v$	$2.07 \log_2 v$
womcodes (incremental or not)	$w \approx t \log_t v$	$1.29 \log_2 v$	$1.55 \log_2 v$

We thank our graduate students Beveraggi, Assaraf and Luguern for their helpful comments.

References.

- [1] R.L. Rivest and A. Shamir, "How to Reuse a "Write-Once" Memory", Inform. and Control 55, 1-19(1982).
- [2] E. Sperner, "Ein Satz uber Untermengen einer endlichen Menge." (1928), Math. Z. 27. 544-548.
- [3] G. D. Cohen et A. Lempel, "Linear Intersecting Codes", To appear in Discrete Mathematics (1985) vol.56(1), pp.35-43.
- [4] A. Fiat et A. Shamir, "Generalized Write-Once Memories", IEEE Trans. on Inform. Theory, Vol. IT-30, No3, pp. 470-480, may 1984.
- [5] G. D. Cohen et P. Godlewski, "Authorized writing for "write-once" memories", Eurocrypt'85, April 9-11, 1985. To appear in "Springer Lecture Notes in Computer Science".
- [6] E.L. Leis, "Data Integrity in Digital Optical Disks", IEEE Trans. on Computers, vol.C-33, pp.818-827, September 1984.
- [7] T.M. Cover, "Enumerative Source encoding", IEEE Trans. on Inform. Theory, vol. IT-19, pp.73-77, January 1973.
- [8] J.M. Berger, "A note on Error Detection codes for Asymmetric Channel", Information and Control 4, pp.68-73, 1961.

How to Reduce your Enemy's Information (extended abstract)[†]

Charles H. Bennett

IBM T. J. Watson Research Laboratory ¹
Yorktown Heights
NY 10598

Gilles Brassard ²

Dépt. IRO, Université de Montréal ³
C.P. 6128, Succ. "A", Montréal
Québec, H3C 3J7

Jean-Marc Robert ⁴

Génie Electrique, Ecole Polytechnique
C.P. 6128, Succ. "A", Montréal
Québec, H3C 3A7

1. INTRODUCTION

In this paper, we investigate how a channel with perfect authenticity but no privacy can be used to repair the defects of a channel with imperfect privacy but no authenticity. More precisely, let us assume that Alice and Bob wish to agree on a secret random bit string. In order to achieve this goal, they have at their disposal an imperfect private channel and an authenticated public channel. The private channel is imperfect in various ways: transmission errors can occur, and partial information can leak to Eve, the eavesdropper, who also can modify the transmissions arbitrarily, as explained below. The only thing Eve cannot do is learn the *entire* contents of the original message sent by Alice. An interesting example of imperfect private "channel", used to exchange (not so random) strings, is Diffie and Hellman's public key distribution scheme [DH], which leaks partial information, even if the discrete logarithm is indeed hard to compute, because it is always feasible for an eavesdropper to determine whether the resulting secret is a quadratic residue or not. The quantum channel [BB1,BB2] is also susceptible to a limited amount of information leakage.

We allow Eve to toggle bits of her choice on the private channel transmissions, or jumble them around, even if she cannot actually read them. This could occur, for instance, if privacy were attempted by enciphering the individual bits with a one-time pad or with a probabilistic encryption scheme [GM] (to toggle an encoded bit, it suffices to multiply its code by the public quadratic non-residue), or alternatively, if a quantum channel were used (by passing selected photons through an appropriate sugar solution). Eve can also suppress the transmission of selected bits and replace them by bits of her choice.

[†] A full paper was submitted for publication in SIAM J. Comp. as *Privacy Amplification Through Public Discussion*.

1. Present address: Boston University, 111 Cummington Street, Boston, MA 02215.

2. Partially supported by NSERC grant A4107 and by NSF grant MCS-8204506.

3. Part of this research was conducted while this author was visiting the University of California, Berkeley.

4. Partially supported by NSERC grant A4107.

On the other hand, the public channel transmits information accurately (possibly because it is supplemented by a classical error-correcting code [MS]), and these transmissions cannot be modified or suppressed by Eve, but their entire contents becomes known to her. The authentication capability can either be enforced by physical properties of the channel or through the use of a universal hashing based authentication scheme [WC]. In the latter case, a small number of random secret bits must be shared initially between Alice and Bob, and some of them can be used only once, so that the net effect of the protocol can be viewed as key expansion rather than key distribution. Computationally secure authentication [Br,GGM] can also be used if protection against unlimited computing power is not sought. We shall assume throughout that Alice and Bob did not share initially any secret information, except perhaps for this public channel authentication feature.

It is instructive to compare our setting with the problem solved by the wire-tap channel of Wyner [W], which achieves similar results in a more classically information-theoretic setting. In Wyner's setting, Alice encodes information by a channel code of her choice. The output of her encoder is fed into two classic (discrete, memoryless) communications channels: the *main channel*, leading to the intended receiver Bob, and the *wire-tap channel*, of lesser capacity than the main channel, leading to the eavesdropper. All participants know the channel code and the statistical properties of both channels. Under these conditions, Wyner showed that by appropriate choice of the channel code, Alice can exploit the difference in capacity between the two channels to communicate reliably with Bob while maintaining almost perfect secrecy from the eavesdropper. In our setting, the users have an additional resource: the authenticated public channel. This allows them to cope with a more powerful eavesdropper. Our eavesdropper is more powerful in two ways, either of which would be fatal in Wyner's setting: she can tamper with Alice's communications as well as listen to them, and she eavesdrops by evaluating an N -bit to K -bit function of her choice, unknown to Alice and Bob, as we shall see in Section 4.2.

In this paper, we assume that some random bit string has already been transmitted from Alice to Bob over the private channel. We investigate authenticated public channel protocols that, with high probability, detect tampering and transmission errors. Subsequent protocols transform both strings in such a way as to eliminate most, and in some cases all, of Eve's information on the resulting string, except for its length. These public channel protocols remain secure against unlimited computing power. Although excessive tampering on the private channel can result in suppressing communications between Alice and Bob, it cannot fool them into thinking that they share a secret random string when in fact their strings are different or otherwise compromised.

This extended abstract contains no proofs and only a selection of the results found in the complete paper [BBR]. For easier reference, we retain here the full paper's numbering for sections, theorems, etc.. In Section 2, we explain why classical error-correcting codes are inappropriate in this context. In Section 3, we investigate how transmission errors and tampering can be detected with high probability, and sometimes corrected, at the cost of leaking some information to Eve. In Section 4, we investigate how Alice and Bob can subsequently reduce arbitrarily Eve's information at the cost of reducing slightly the length of their shared random string, assuming they have an *a priori* upper bound on the amount of information collected by Eve on the private channel. In Section 5, we investigate the possibility of depriving Eve entirely from any information on the final shared random string, at the cost of reducing its length more substantially.

Before we get started, let us give the following definition and some notation: if $i < j$, a function $f: \{0,1\}^j \rightarrow \{0,1\}^i$ is *equitable* if $\#\{x \mid f(x) = a\} = 2^{j-i}$ for every binary string a of length i . If x and

y are equal length bit strings, $x \oplus y$ denotes their bit-by-bit exclusive-or. Finally, if x is a length N bit string and if $0 \leq K \leq N$, $x \bmod 2^K$ denotes the length K bit string consisting of the rightmost K bits of x , and $x \operatorname{div} 2^K$ denotes the length $N-K$ bit string obtained from x by deleting its rightmost K bits. We shall herein assume that the reader is familiar with the classical notions of error-correcting codes [MS], information theory [G], universal hashing [CW,WC], and the theory of finite fields [Be].

2. THE INADEQUACY OF CLASSICAL ERROR-CORRECTING CODES

Let us recall that the imperfect private channel considered here is susceptible, not only to random transmission errors, but also to any amount of controlled tampering. The classical theory of error-correcting codes [MS], on the other hand, is based on the assumptions that few errors are more likely to occur than many, and that errors are not maliciously set by an opponent. It is therefore not quite adequate for our purpose.

For instance, let x and y be Alice and Bob's strings, respectively, and let N be their length. Eve's ability to toggle bits of her choice enables her to actually select $x \oplus y$, barring actual transmission errors. This is clearly intolerable if error detection is attempted through a linear error-correcting code [MS]. Indeed, let x be the private channel transmitted codeword corresponding to Alice's chosen random string. Let z be any codeword chosen by Eve. If she perturbs the private channel transmission so that Bob receives $y = x \oplus z$, it will not be possible for him to detect tampering. Notice that Eve can achieve this without gaining any knowledge on the contents of the original transmission x .

Should Alice randomly shuffle the codeword bits, in an attempt to preclude this threat, and publicly tell Bob how to unscramble them only after the private channel transmission is completed, it would no longer be possible for Eve to toggle selected bits and be certain to escape detection. However, if a Hamming code of dimension $[N,K]$ is used, for instance, Eve can toggle 3 random bits and escape detection with probability $1/(N-2)$. Using such a protocol, Alice and Bob could only achieve a high probability of not being fooled, say $1 - 2^{-50}$, at the cost of exchanging unreasonably long strings. In Section 3.1, we describe error detection schemes such that the probability of undetected tampering and transmission errors is independent of the number and position of altered bits. Moreover, this probability can be exponentially small in the length of the strings transmitted.

3. DETECTION AND CORRECTION OF TRANSMISSION ERRORS AND TAMPERING

Let x be some random bit string selected by Alice. Assume she transmits it directly through the imperfect private channel, and let y be the string thus received by Bob. Let N be the length of both strings. We investigate public channel protocols that allow Alice and Bob to detect whenever $x \neq y$ with an arbitrarily small error probability, independently of how y differs from x . The fact that these protocols leak information to Eve about x is considered in Section 4.

3.1. Error detection

A very simple but impractical way of testing whether $x = y$ is for Alice to choose a random function $f: \{0,1\}^N \rightarrow \{0,1\}^K$, where K is a security parameter. After the private channel transmission is completed, she sends $f(x)$ to Bob over the public channel, together with a complete description of the function f . Should Bob find out that $f(y) = f(x)$, this would be considered as strong evidence that $y = x$, the error probability being 2^{-K} . On the other hand, should $f(y)$ be different from $f(x)$, Bob

could report to Alice with certainty that he did not receive the correct string. The amount of information on x leaking to Eve from this protocol depends only on the security parameter K , and not on the length N of the strings (except of course for the fact that $K < N$). This would not be the case if a classical error-detecting code had been used. Unfortunately, this scheme cannot be used in practice because there are too many such functions, so that $K2^N$ bits are typically needed to merely transmit a description of the randomly chosen function.

Universal hashing [CW] provides an efficient way to achieve the same goal. After the private channel transmission is completed, Alice randomly chooses a function $f: \{0,1\}^N \rightarrow \{0,1\}^K$ among some standard universal₂ class of functions. She then sends both $f(x)$ and a description of f to Bob. Thanks to universal hashing, the description of f can be transmitted efficiently. After computing $f(y)$, Bob checks whether it agrees with $f(x)$. If it does, a basic property of universal hashing allows them to assume that $x = y$, their probability of error being bounded by 2^{-K} .

3.2. Reconciliation of the strings

Whether $f: \{0,1\}^N \rightarrow \{0,1\}^K$ is chosen as a completely random function or within some universal₂ class of functions, what should Alice and Bob do whenever $f(x)$ differs from $f(y)$? If the private channel is reliable enough that only one or perhaps two errors are to be expected at most, it may be worthwhile for Bob to try computing $f(z)$ on all strings z differing from y by only one bit or two, in the hope of finding a match with $f(x)$ and thus a likely candidate z for x .

If many transmission errors are to be expected, this would be much too time consuming. In the full paper, we offer two different solutions to this problem, one based on the *post-facto* application of a convolutional code and one based on a blockwise exclusive-or strategy. The effect of the convolutional code protocol is to allow Bob to transform y into x with high probability, at the cost of disclosing to Eve some information about x . Protocols from Section 4 can subsequently be applied to reduce that information. On the other hand, the effect of the exclusive-or strategy is to transform both x and y into a probably common shorter string z on which Eve has no more information than she initially had on x from eavesdropping over the private channel.

4. REDUCTION OF THE EAVESDROPPER'S INFORMATION

Assuming that Alice and Bob agree on their strings as a result of one of the protocols discussed above, Eve has two different sources of information on that string: deterministic information obtained from eavesdropping on the private channel, as the original random bit string was being transmitted, and stochastic information resulting from eavesdropping on the public channel, as the agreement protocol was being carried out.

In this section, we investigate how to reduce Eve's information arbitrarily close to zero, at the cost of slightly shrinking the random bit string shared between Alice and Bob at the end of the protocol. In a first step, we assume that no eavesdropping on the private channel has occurred, but that tampering and transmission errors were possible. In a second step, we assume to the contrary that a limited amount of eavesdropping on the private channel is susceptible of having occurred, but that it is not necessary to carry out an agreement protocol from Section 3, thus depriving Eve from this potential stochastic information. Finally, the full paper considers the case where both sources of information are simultaneously available to her. All these protocols are secure against an eavesdropper with unlimited computing power.

4.1. Reducing the public channel eavesdropper's information

Let us assume for the moment that Eve did not attempt eavesdropping on the private channel, but that she has complete information on the error detection protocol carried out between Alice and Bob over the public channel. Let x be the random string of length N on which Alice and Bob have just agreed, and let $f: \{0,1\}^N \rightarrow \{0,1\}^K$ be their error detection function. Eve knows the K -bit value of $f(x)$, together with the function f itself. Her information can be characterized by the set $C = \{z \in \{0,1\}^N \mid f(z) = f(x)\}$ of possible candidates for x . From Eve's point of view, each element of C is equally likely to be the string x currently shared between Alice and Bob. Notice that Alice and Bob also have complete knowledge on the set C .

In order to reduce Eve's information, Alice and Bob publicly agree on a function $g: \{0,1\}^N \rightarrow \{0,1\}^R$, for some integer $R \leq N-K$, such that knowledge of the set C gives arbitrarily little information on $g(x)$, or perhaps even none at all. The final string on which Alice and Bob agree is thus $g(x)$. In other words, the purpose of this function g is to shrink the string x by at least K bits, in order to compensate for the K bits of information that knowledge of C gives Eve.

4.1.1. The case of truly random functions

Assume the error detection function f was chosen randomly among all N -bit to K -bit functions. Let $g: \{0,1\}^N \rightarrow \{0,1\}^R$ be the function $g(x) = x \bmod 2^R$. Let $S = N-K-R$, then

Theorem 8. The expected amount of information known by Eve on $g(x)$ from knowledge of f , g and $f(x)$ is less than $2^{-S}/\ln 2$ bit.

Here, S should be thought of as the number of additional bits sacrificed to privacy. Sacrificing one more bit in the final string chops in half Eve's information about it. This holds even if Eve knows in advance which information reduction function g is to be used. Any other equitable N -bit to R -bit function would have performed just as well.

4.1.3. The case of universal hashing

Let us now assume that a practical error detection protocol was used: the function $f: \{0,1\}^N \rightarrow \{0,1\}^K$ was randomly chosen among some universal₂ class of hash functions. Rather than developing a general theory of information reduction in this context, let us design an *ad hoc* technique for a given universal₂ class.

Let a and b be elements of $\text{GF}(2^N)$ [Be] such that $a \neq 0$. The degree one polynomial $q_{a,b}(x) = ax + b$, arithmetic being done in $\text{GF}(2^N)$, defines a permutation of $\text{GF}(2^N)$. If we let $\sigma: \{0,1\}^N \rightarrow \text{GF}(2^N)$ stand for the natural one-one correspondence, this induces a permutation $\pi_{a,b}: \{0,1\}^N \rightarrow \{0,1\}^N$ defined by $\pi_{a,b}(x) = \sigma^{-1}(q_{a,b}(\sigma(x)))$. Therefore, for any fixed $K \leq N$, the function $h_{a,b}: \{0,1\}^N \rightarrow \{0,1\}^K$ defined by $h_{a,b}(x) = \pi_{a,b}(x) \bmod 2^K$ is equitable. Furthermore, the class of all such functions $h_{a,b}$, for every $a, b \in \text{GF}(2^N)$, $a \neq 0$, forms a universal₂ class of hash functions, so that it can be used for the error detection protocol.

Theorem 15. Let a and b be any elements of $\text{GF}(2^N)$ such that $a \neq 0$. Let x be a random string of length N . Then knowledge of a , b and $h_{a,b}(x)$ gives no information on the length $N-K$ string defined as $\pi_{a,b}(x) \text{ div } 2^K$.

Use of this universal₂ class allows Alice and Bob to verify whether their strings are identical, with a probability of error at most 2^{-K} . If they turn out to be the same, they can be transformed into a new string that is only K bits shorter, on which Eve has no information at all. This is optimal.

4.2. Reducing the private channel eavesdropper's information

Let us now assume that partial eavesdropping has occurred on the private channel. Let K be an upper bound on the number of bits of information thus obtained by Eve, where $K < N$. This can be formalized as follows in general: Eve chooses any function $e : \{0,1\}^N \rightarrow \{0,1\}^K$, and she obtains the value of $e(x)$ after x has been transmitted over the private channel. Of course, Alice and Bob have no information on which function e was chosen by Alice, except for an upper bound on K .

The effect of eavesdropping over the private channel is very similar to that of eavesdropping over the public channel, as described in Section 3, in that the information gained by Eve can be characterized by a set $E = \{z \in \{0,1\}^N \mid e(z) = e(x)\}$ of possible candidates for x . However, there is a fundamental difference: it is no longer true that Alice and Bob have complete knowledge on E . For this reason, it is not possible for them, in general, to eliminate Eve's information with certainty.

Theorem 17. No matter how Alice and Bob choose their function $g : \{0,1\}^N \rightarrow \{0,1\}^R$, for any $R > 0$, there always is an equitable function $e : \{0,1\}^N \rightarrow \{0,1\}^K$, for any $K > 0$, such that knowledge of e , g and $e(x)$ yields information on $g(x)$.

Therefore, the best Alice and Bob can hope for is to reduce arbitrarily Eve's information. There can be no analogue to Theorem 15. Nonetheless, if we restrict even further Eve's choice of e , so that she can only read a selection of K physical bits of x , it becomes possible again for Alice and Bob to eliminate her information entirely, as discussed in Section 5.

For simplicity, let us assume that transmission errors and tampering are not a worry for Alice and Bob, so that an error detection protocol is not carried out. This assumption is removed in Section 4.3 of the full paper. Let x be the length N bit string common to Alice and Bob, and let $e(x)$ be the K -bit information known by Eve about x . Alice and Bob wish to publicly agree on some function $g : \{0,1\}^N \rightarrow \{0,1\}^R$, for some $R \leq N-K$, such that knowledge of e , $e(x)$ and g leaves Eve with an arbitrarily small fraction of one bit of information about $g(x)$.

Here again, we consider two approaches for the reduction of Eve's information: one based on truly random functions and one based on universal hashing techniques. The first approach is only of theoretical interest, but the second one is efficient in practice.

4.2.1. The case of truly random functions

Theorem 19. Let $e : \{0,1\}^N \rightarrow \{0,1\}^K$ be any function, let $S < N-K$ be a security parameter, and let $R = N-K-S$. If $g : \{0,1\}^N \rightarrow \{0,1\}^R$ is chosen randomly, the expected amount of information on $g(x)$ given by knowledge of e , g and $e(x)$ is at most $2^{-S}/\ln 2$ bit.

4.2.2. The case of universal hashing

Contrary to the error detection protocols of Section 3, it is no longer sufficient to consider universal₂ classes: here, we use *strongly* universal₂ classes [WC].

Theorem 21. Let e , S and R be as in Theorem 19, let H be a publicly known strongly universal₂ class of hash function from $\{0,1\}^N$ to $\{0,1\}^R$ and let g be a function chosen randomly within H . The expected amount of information on $g(x)$ given by knowledge of e , g and $e(x)$ is at most $2^{-S}/\ln 2$ bit.

The above theorem is true despite the fact that Eve already knows the class H , but of course not the specific function g , when she gets to choose her function e .

5. ELIMINATION OF THE EAVESDROPPER'S INFORMATION

The protocols of Section 4.2 should be sufficient for most applications, despite the fact that Eve still has an arbitrarily small fraction of one bit of information on the resulting shared random string. Although we were able to eliminate her information entirely in Theorem 15, the techniques used could only be applied because Alice and Bob had complete knowledge of Eve's information. As shown in Theorem 17, this cannot be extended whenever Eve is allowed to access information of *her choice* from the private channel transmission.

In this section, we investigate a protocol by which Alice and Bob can nonetheless wipe out Eve's information, assuming that she obtained a maximum of K *physical* bits of her choice from the private channel transmission. Although the value of K is known to Alice and Bob, they do not know, of course, which particular bits of their string are compromised. This protocol is expensive in the sense that the resulting string is generally substantially shorter than those resulting from the protocols of Section 4.2; however, this is the unavoidable price to pay in order to make sure that Eve is left with no information at all.

5.1. The notion of (N, J, K) -functions

For any integers N , J and K such that $N \geq J+K$, $J > 0$ and $K > 0$, a function $f: \{0,1\}^N \rightarrow \{0,1\}^J$ is said to be (N, J, K) if, no matter how one fixes any K of its input bits, each of the 2^J output bits can be produced in exactly 2^{N-J-K} different ways by varying the remaining $N-K$ input bits. Intuitively, an (N, J, K) -function compresses an N bit string into a J bit string in such a way that knowledge of any K of the input bits gives no information on the output. This is equivalent to the notion of t -resilient functions independently introduced by [CGHFRS].

Given such a function, Alice and Bob can apply it to their respective strings, thus producing a new (shorter) string on which Eve has no information. Notice that this still holds even if she already knows which function will be used by Alice and Bob in advance of her deciding which K bits to read from the private channel. Therefore, the subsequent public transmission between Alice and Bob is not necessary in this case, as it can be replaced by a standard protocol.

The case $J = N-K$ is the best possible because there is no hope to produce a completely secret string of length $N-K+1$ if Eve knows K of the original N bits. A function f that is $(N, N-K, K)$ is said to be (N, K) . The following theorem shows how to build (N, K) -functions whenever they exist.

Theorem 23.

- 1) For any $N > 1$, there are $(N, 1)$ and $(N, N-1)$ -functions.
- 2) For any $N > 3$, there are no (N, K) -functions whenever $1 < K < N-1$.

5.2. How to build (N, J, K) -functions

We wish to answer the following question: given N and K , what is the maximum value for J such that an (N, J, K) -function exists? In other words, what is the longest secret random string on which Alice and Bob can agree if they start from a random string of length N , of which K bits are compromised. Theorem 23 shows that J must be strictly smaller than $N-K$ unless $K = 1$ or $K = N-1$.

We were unable to answer the above question in its full generality. For this reason, we restrict our attention to the special class of (N, J, K) -functions for which every output bit is produced as the exclusive-or of some of the input bits. Such functions are referred to as $\text{xor-}(N, J, K)$ -functions. We conjecture that these functions are as efficient as possible, in the sense that if no $\text{xor-}(N, J, K)$ -functions exist for given values of N, J and K , then no general (N, J, K) -functions exist either. This *Xor-Conjecture* is proved in [CGHFRS] for the case $J = 2$, but it is not believed in general by all members of [CGHFRS].

The following characterization, known as the *Xor-Lemma*, allows to establish an equivalence between $\text{xor-}(N, J, K)$ -functions and binary linear codes [MS].

Lemma 25 (independently discovered by [CGHFRS]). Let M be a $J \times N$ Boolean matrix. Let $f: \{0,1\}^N \rightarrow \{0,1\}^J$ be the function represented by M in the natural way (i.e. $f(x)^t = Mx^t$, all operations being performed modulo 2). The function f is (N, J, K) if and only if the exclusive-or of any non-empty set of rows of M contains at least $K+1$ ones.

The equivalence is now stated:

Theorem 26 (independently discovered by [CGHFRS]). For given values of N, J and K , there exists an $\text{xor-}(N, J, K)$ -function if and only if there exists an $[N, J]$ binary linear code with minimum distance at least $K+1$ between any two codewords.

Consequently, our problem is equivalent to a classical problem of algebraic coding theory. Unfortunately, no efficient algorithms are known, much less closed formed formulae, to determine the largest possible minimum codeword distance among all $[N, J]$ binary linear codes. There are, however, several classical lower and upper bounds on this value [MS], and these bounds apply just as well to our problem.

For instance, Hamming codes tell us that $\text{xor-}(2^L-1, 2^{L-1}-1, 2)$ -functions exist for every $L \geq 2$. Conversely, Hamming's upper bound show that no $\text{xor-}(2^L-1, 2^{L-1}-1, 2)$ -functions can exist. Elimination of Eve's information in this case ($K=2$) costs $L-2-S$ more bits than if we had been satisfied to reduce her information below $2^{-S}/\ln 2$ bit, as in Section 4.2. Similarly, Griesmer's upper bound and the simplex code allow to build $\text{xor-}(2^L-1, L, 2^{L-1}-1)$ -functions for any $L \geq 2$, whereas neither $\text{xor-}(2^L-1, L, 2^{L-1})$ -functions nor $\text{xor-}(2^L-1, L+1, 2^{L-1}-1)$ -functions can exist. Finally, Varsharmov-Gilbert's lower bound together with McEliece's upper bound allow to construct $\text{xor-}(N, J, K)$ -functions such that J is at least half the optimal (xor) value, as long as $K/N < 0.3$ and N is large enough. We encourage the reader to consult [CGHFRS] for additional results on (N, J, K) (*alias* t -resilient) functions.

6. CONCLUSIONS

If no eavesdropping occurred over the private channel, it is possible for Alice and Bob to publicly verify that no transmission errors nor tampering occurred either, with a 2^{-K} error probability,

and end up with an entirely secret final string that is only K bits shorter than the original private transmission. This is optimal. A somewhat shorter common string, on which Eve still has no information, can also be obtained with high probability despite transmission errors over the private channel.

If partial eavesdropping occurred over the private channel, leaking up to K bits of information to Eve, in Shannon's sense, it is still possible for Alice and Bob to publicly verify that no transmission errors nor tampering occurred, with a 2^{-L} error probability, and end up with a final string that is $K+L+S$ bits shorter than the original private transmission, on which Eve has less than $2^{-S}/\ln 2$ bit of information. Here again, transmission errors can be handled at the cost of reducing some more the length of the final common string.

Finally, if partial eavesdropping over the private channel is restricted to K physical bits secretly chosen by Eve, it becomes possible again for Alice and Bob to verify with high probability that no errors nor tampering occurred, and end up with a new string on which Eve has no information whatsoever. However, the new string is substantially shorter than if Alice and Bob had tolerated knowledge by Eve of an arbitrarily small fraction of one bit of information.

7. REFERENCES

- [Be] E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, 1968.
- [Br] G. Brassard, "On Computationally Secure Authentication Tags Requiring Short Secret Shared Keys", in *Advances in Cryptology: Proc. of Crypto 82*, D. Chaum, R. L. Rivest and A. T. Sherman, eds., Plenum, New York, 1983, pp. 267-275.
- [BB1] C. H. Bennett and G. Brassard, "Quantum Cryptography and its Application to Provably Secure Key Expansion, Public-Key Distribution and Coin-Tossing", in *IEEE International Conference on Computers, Systems and Signal Processing*, Bangalore, India, December 1984, pp. 175-179.
- [BB2] C. H. Bennett and G. Brassard, "An Update on Quantum Cryptography", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp. 475-480.
- [BBR] C. H. Bennett, G. Brassard and J.-M. Robert, "Privacy Amplification through Public Discussion", submitted to *SIAM J. Comput.*, 1985.
- [CW] J. L. Carter, and M. N. Wegman, "Universal Classes of Hash Functions", *J. Comput. System Sci.*, 18 (1979), pp. 143-154.
- [CGHFRS] B. Chor, O. Goldreich, J. Hastad, J. Freidmann, S. Rudich and R. Smolensky, "The Bit Extraction Problem or t-Resilient Functions", in *Proc. 26th IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1985, pp. 396-407.
- [DH] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Trans. Information Theory*, IT-22 (1976), pp. 644-654.
- [G] R. G. Gallager, *Information Theory and Reliable Communication*, John Wiley and Sons, New York, 1968.
- [GGM] O. Goldreich, S. Goldwasser and S. Micali, "How to Construct Random Functions", in *Proc. 25th IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1984, pp. 464-479.
- [GM] S. Goldwasser and S. Micali, "Probabilistic Encryption". *J. Comput. System Sci.*, 28 (1984), pp. 270-299.
- [MS] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, New York, 1977.
- [WC] M. N. Wegman and J. L. Carter, "New Hash Functions and Their Use in Authentication and Set Equality", *J. Comput. System Sci.*, 22 (1981), pp. 265-279.
- [W] A. D. Wyner, "The Wire-Tap Channel", *Bell System Journal*, 54 (1975), pp. 1355-1387.

Encrypting Problem Instances

Or . . . , Can You Take Advantage of Someone
Without Having to Trust Him?

Joan Feigenbaum*

Computer Science Department
Stanford University
Stanford, CA 94305

1. Introduction

This paper describes ongoing work on the task of *encrypting problem instances*, also known as *computing with encrypted data*. A *problem* is specified by a function f and an *instance* by a value x in the domain of f . The scenario involves two people, A and B. A has instances $\{x_i\}$ of f to which she needs answers, but she lacks the resources to compute them. We use the term resources completely generally—she may be lacking time, space, algorithmic knowledge, or appropriate hardware, or she may simply be too lazy to implement a solution that she knows others have already implemented. B has the resources to compute $f(x)$ and is willing to let A use them, i.e., he is willing to send her $f(x)$ if she sends him x . She would like to take advantage of his generosity without having to trust him, i.e., she does not want to reveal any more about her data than she must in order to enable him to compute the correct answer. Intuitively, we say that f is *encryptable* if A can easily transform instance x into instance x' , obtain $f(x')$ from B, and easily compute $f(x)$ from $f(x')$ in such a way that B cannot infer x from x' .

* The author did some of this work while at AT&T Bell Laboratories for the summer. During the academic year, she is funded by a Xerox Corporation Fellowship and a grant from the AT&T Bell Laboratories Graduate Research Program for Women.

In the commutative diagram of Figure 1, the horizontal arrows $x \rightarrow x'$ and $f(x') \rightarrow f(x)$ represent computations done by A, and the vertical arrows $x \rightarrow f(x)$ and $x' \rightarrow f(x')$ computations that only B can do. B actually does the computation $x' \rightarrow f(x')$ but not the computation $x \rightarrow f(x)$ because A does not want him to know x . The instance x is called the *cleartext instance* and the instance x' the *encrypted instance*.

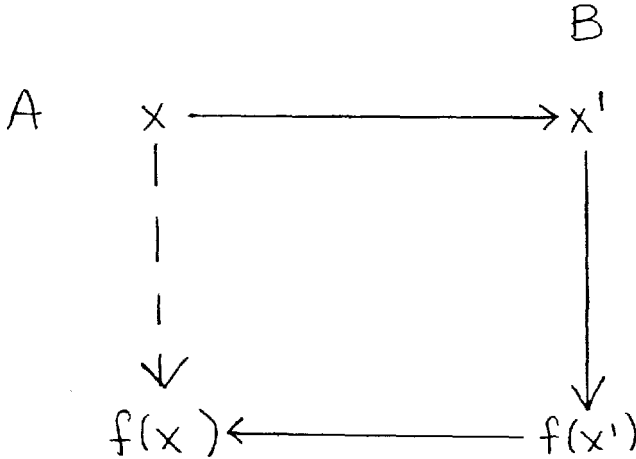


Figure 1. Because the diagram commutes, A learns the value of $f(x)$. A does the inexpensive computations $x \rightarrow x'$ and $f(x') \rightarrow f(x)$. B does the expensive computation $x' \rightarrow f(x')$.

What follows is an attempt to formalize the statement “B cannot infer x from x' ” and to give important examples of encryptable problems. Under one plausible definition of encryptability, all NP-complete problems that are P-isomorphic to CNF-SAT are encryptable.

It is important to keep the following aspects of the problem clear:

1) A’s mistrust of B is confined to her fear that he will do something objectionable with her data; that is, she *does* trust him to give her the right answer $f(x')$ to her encrypted instance.

2) Encryptability is a property of the problem f , not of a particular solution to it. Thus, in her search for an encryption scheme, A cannot make assumptions about B’s algorithm for computing $f(x')$.

3) A is not searching for a public key encryption algorithm. The security of x may rest on the fact that probabilistic choices she makes during the computation $x \longrightarrow x'$ are made in secret.

2. Motivating Examples

Example 1 is a scheme for encrypting instances of the discrete logarithm problem. As we go on, we will only consider formal definitions of encryptability that accept the scheme in Example 1, because it very clearly meets A's needs.

Example 1: Fix a large prime p and a generator g for the cyclic group Z_p^* . For $x \in Z_p^*$, A wishes to find the unique exponent $e = f(x)$ in $\{1, \dots, p-1\}$ such that $g^e \equiv x \pmod{p}$. To encrypt the instance x , she chooses a random element c of $\{1, \dots, p-1\}$ and lets

$$x' \leftarrow x \cdot g^c \pmod{p}.$$

She sends x' off to B, and decrypts the answer by computing

$$f(x) \leftarrow f(x') - c \pmod{p-1}.$$

This scheme is obviously *feasible*: Simple arithmetic shows that she will get the correct value for the discrete logarithm of x , and both computations $x \cdot g^c \pmod{p}$ and $f(x') - c \pmod{p-1}$ can be done in time polynomial in $\log p$. More importantly, it is also *secure*, because, for any pair of values x and x' in $\{1, \dots, p-1\}$, there is an exponent c such that $x' \equiv x \cdot g^c \pmod{p}$. Thus we can say quite literally that, without knowing c , B cannot infer x from x' .

Example 2 is a scheme for encrypting instances of the clique problem that I'd like to reject because it's not secure. The analysis of Example 2 will point us to a precise definition of security. The scheme uses the following definition of graph multiplication: If G and H are undirected graphs without self-loops, then $G[H]$ (" G composed with H ") is a finite graph with $V(G[H]) = V(G) \times V(H)$ and $E(G[H]) = \{(x, y) - (v, w) : x - v \in E(G) \text{ or } x = v \text{ and } y - w \in E(H)\}$. By the *clique number* of a graph G , we mean the number k such that G has a clique of size k but no clique of size $k+1$.

Example 2: A's instance x is a pair G, k , where G is a finite, undirected graph without self-loops and k is an integer between 1 and $|V(G)|$, about which she inquires "does G have a complete subgraph with k nodes"? To encrypt x , A chooses a random graph H with clique-number j and computes

$$x' \leftarrow G[H], kj.$$

The answer to x' that she gets from B is the same as the answer to x , as the following lemma shows. Thus, the decryption step $f(x') \rightarrow f(x)$ is trivial in this scheme.

Lemma 1: If H has clique number j , then G has a clique of size k if and only if $G[H]$ has a clique of size kj .

Proof: If $\{v_1, \dots, v_k\}$ is a clique of G and $\{w_1, \dots, w_j\}$ is a clique of H , then $\{(v_a, w_b), 1 \leq a \leq k, 1 \leq b \leq j\}$ is a clique of $G[H]$. Conversely, if $C = \{x_1, \dots, x_{kj}\}$ is a clique of $G[H]$, then no more than j nodes in C can lie in a single copy of H . So C intersects at least k copies of H . Between any pair of these copies, there is at least one edge, and so all possible edges are present. Hence these copies correspond to at least k nodes of G that form a clique. ■

The scheme in Example 2 is also feasible: A can grow H from one or more cliques of size j by adding only nodes of degree less than j , and she can construct $E(G[H])$ straightforwardly in time $O(|E(G)| \cdot |E(H)|)$. But we claim that the scheme is not secure. It is insecure because of the small number of possible cleartext instances that can correspond to an encrypted instance G', k' . Coppersmith and Feigenbaum show in [CF] that most composite graphs G' can be written as $G[H]$ for only one pair of graphs G, H and that if a graph has two inequivalent factorizations $G_1[H_1] \cong G_2[H_2]$, then $|V(G_1)| > |V(G_2)|$. Thus the number of possible cleartext instances that can be encrypted as G', k' can be very generously upper-bounded by the number of integer factorizations of $|V(G')|$ times the number of integer factorizations of k' , which is all polynomial in $|V(G')|$. In the rare cases in which B cannot infer a unique G for which $G[H] \cong G'$, he can at least infer that G is a member of a small set.

But *how* can he infer this information? In other words, what is the complexity of factoring graphs under this definition of multiplication? Feigenbaum and Schäffer have shown that it is the same, to within polynomial factors, as the complexity of testing whether two connected graphs are isomorphic [FS].

Because there is no known polynomial-time algorithm for testing graph isomorphism, one is tempted to say that graph multiplication is a one-way function and hence this scheme is secure. Recall, however, that B is solving instances of the clique problem. So, unless $P = NP$, he *has* more than polynomial time and could decide to spend it decrypting x' rather than solving it.

In this crucial way, our version of computing with encrypted data departs from recent work on cryptography by the theoretical computer science community. We want to say what it means to encrypt instances of *hard* problems, for which B has to be given a lot of time, and hence cannot allow schemes whose security rests on intractability *assumptions*. Rather than saying, as has been the fashion in computer science, that the cryptanalyst cannot decrypt the instance he sees because he does not have enough *time*, we want to return to more conventional criteria in cryptography and say that he cannot decrypt it because he does not have enough *information*. This is the case in Example 1, where B cannot figure out anything interesting because he does not know which value of c was used in computing x' .

3. A Precise but Lenient Definition of Encryptability

In this section, we explore the consequences of the lesson of Example 2. The graph-composition scheme fails because the number of cleartext instances that correspond to a given encrypted instance is too small. In the following definition of a successful encryption scheme, this situation is precluded explicitly.

Suppose for now that f is a decision problem. In her encryption algorithm E , A will combine elements of $Dom(f)$ with *keys* drawn from some convenient set K . The nature of K depends on the problem f . In Example 1, K was the set of exponents $\{1, \dots, p-1\}$.

Definition 1: $E : Dom(f) \times K \rightarrow Dom(f)$ is a successful encryption function for the decision problem f if:

- 1) $E(x, k)$, $x \in Dom(f)$, $k \in K$, can be computed in time polynomial in $|x|$,
- 2) $E(x, k)$ is a yes-instance of f if and only if x is a yes-instance of f ,

3) If x' is in the range of E , then

$$|\{x: \exists k \in K: E(x, k) = x'\}| \neq O(p(|x'|))$$

for any polynomial p , and

4) If $E(x, k_0) = x'$ for a particular key k_0 , then

$$|\{k: E(x, k) = x'\}| = O(q(|x'|))$$

for some polynomial q .

Conditions 1 and 2 ensure that the encryption scheme is feasible; in fact condition 2 eliminates the need to do any decryption $f(x') \rightarrow f(x)$. The moral of Example 2 is embodied in condition 3, which says that the number of cleartext instances in the preimage of a particular encrypted instances x' is superpolynomial is the size of the instances. Condition 4 is a technical requirement for security: Say $x' \in \text{Range}(E)$, $|x'| = n$, and $\{x_0, \dots, x_{2^n-1}\}$ is the complete set of cleartext instances in its preimage. If $\{k_1, \dots, k_{2^n}\}$ is a complete set of the keys that can result in the encrypted instance x' , $E(x_0, k_i) = x'$ for $2^{n-1} + 1 \leq i \leq 2^n$, $E(x_i, k_i) = x'$ for $1 \leq i \leq 2^{n-1}$, and $E(x_i, k_j) \neq x'$ for $1 \leq i \leq 2^{n-1}$ and $i \neq j$, then there are a superpolynomial number of preimages of x' , but they are extremely unequally probable. If A draws keys uniformly from $\{k_1, \dots, k_{2^n}\}$ and happens to wind up with encrypted instance x' , then the probability is at least $\frac{1}{2}$ that she started with cleartext instance x_0 . This is not possible if condition 4 is satisfied. Note that there is no requirement that the function E be surjective.

Example 3 is a scheme for encrypting instances of the Comparative Vector Inequalities (CVI) problem; it is clear that the scheme satisfies conditions 1, 3, and 4, so the proof is omitted. Plaisted showed that CVI is NP-complete [P].

Example 3: Each instance of CVI consists of two sets of m -tuples of integers $\{\bar{x}_1, \dots, \bar{x}_k\}$ and $\{\bar{y}_1, \dots, \bar{y}_l\}$ about which A asks whether there is an m -tuple \bar{z} such that the number of \bar{x}_i satisfying $\bar{x}_i \geq \bar{z}$ is strictly greater than the number of \bar{y}_j satisfying $\bar{y}_j \geq \bar{z}$, where $\bar{u} \geq \bar{v}$ if and only if no component of \bar{u} is less than the corresponding component of \bar{v} . To encrypt an instance, A chooses an element $\bar{w} \in Z^m$ and computes

$$E(\{\bar{x}_1, \dots, \bar{x}_k\}, \{\bar{y}_1, \dots, \bar{y}_l\}, \bar{w}) = \{\bar{x}_1 + \bar{w}, \dots, \bar{x}_k + \bar{w}\}, \{\bar{y}_1 + \bar{w}, \dots, \bar{y}_l + \bar{w}\}.$$

Then \bar{z} satisfies $|\{\bar{x}_i: \bar{x}_i \geq \bar{z}\}| > |\{\bar{y}_j: \bar{y}_j \geq \bar{z}\}|$ if and only if $\bar{z}' = \bar{z} + \bar{w}$ satisfies $|\{\bar{x}'_i: \bar{x}'_i \geq \bar{z}'\}| > |\{\bar{y}'_j: \bar{y}'_j \geq \bar{z}'\}|$, where $\bar{x}'_i = \bar{x}_i + \bar{w}$ and $\bar{y}'_j = \bar{y}_j + \bar{w}$.

Having shown that *one* NP-complete problem admits an encryption scheme satisfying Definition 1, we cannot avoid asking whether they all do. For each NP-complete problem f , there is a polynomial-time reduction r of f to CVI that takes yes-instances to yes-instances and no-instances to no-instances. Can A encrypt an instance x of f by first applying r and then applying the encryption function E from Example 3 to $r(x)$? Not necessarily: The fact that r may not be surjective prevents us from proving that the mapping $E \circ r$ satisfies conditions 3 and 4 of Definition 1.

If r were truly a *structure preserving*, polynomial-time computable mapping, then E could be composed with it to yield an encryption function for f . Berman and Hartmanis consider such a class of mappings, the p -isomorphisms, in [BH]. We will restate one of their results and then use it to prove something general about the encryptability of NP-complete problems. Let Σ and Γ be two alphabets, C a subset of Σ^* , and D a subset of Γ^* . A p -reduction of C to D is a transducer $T: \Sigma^* \rightarrow \Gamma^*$ that runs in polynomial time such that $T(x) \in D$ if and only if $x \in C$. A p -isomorphism is a *bijection* $f: \Sigma^* \rightarrow \Gamma^*$ such that f is a p -reduction of C to D and f^{-1} is a p -reduction of D to C . Note that the functions f and f^{-1} run in polynomial time on $\Sigma^* \setminus C$ and $\Gamma^* \setminus D$ as well as on C and D .

Theorem (Berman and Hartmanis): An NP-complete set U is p -isomorphic to CNF-SAT if and only if there exist two p -time computable functions S_U and D_U such that

- (i) $(\forall x, y) [S_U(x, y) \in U \text{ iff } x \in U]$,
- (ii) $(\forall x, y) [D_U(S_U(x, y)) = y]$.

It is straightforward to find appropriate functions S and D for the CVI problem of Example 3.

Lemma 2: CVI is p -isomorphic to CNF-SAT.

Proof: Suppose $(X = \{\bar{x}_1, \dots, \bar{x}_k\}, Y = \{\bar{y}_1, \dots, \bar{y}_l\})$ is an instance of CVI,

where $\overline{x_i} = (x_{i1}, \dots, x_{im})$ and $\overline{y_j} = (y_{j1}, \dots, y_{jm})$. Then put

$$S_{\text{CVI}}((X, Y), v) = (\{x'_1 = (x_{11}, \dots, x_{1m}, v), \dots, x'_k = (x_{k1}, \dots, x_{km}, v)\}, \\ \{y'_1 = (y_{11}, \dots, y_{1m}, v), \dots, y'_l = (y_{l1}, \dots, y_{lm}, v)\}).$$

We have $S_{\text{CVI}}(X, Y, v) \in \text{CVI}$ if and only if $(X, Y) \in \text{CVI}$ because (z_1, \dots, z_m) is less than or equal to more $\overline{x_i}$'s than $\overline{y_j}$'s if and only if (z_1, \dots, z_m, v) is less than or equal to more $\overline{x'_i}$'s than $\overline{y'_j}$'s. The obvious algorithm for D_{CVI} (scan $\overline{x'_1}$ and return its rightmost component) gives us $D_{\text{CVI}}(S_{\text{CVI}}((X, Y), w)) = w$. ■

Berman and Hartmanis state that they know of no NP-complete problems that are not p -isomorphic to CNF-SAT. In particular, they show that CLIQUE is p -isomorphic to CNF-SAT. The question of how many p -isomorphism classes there are among the NP-complete problems remains open, but Mahaney subsequently proved that the number is either one or countably infinite [M]. By Lemma 2, all the NP-complete problems that have been classified are in the same p -isomorphism class as CVI.

Lemma 3: If f is a decision problem that is p -isomorphic to CVI, then f is encryptable under Definition 1.

Proof: Let E_{CVI} be the encryption function for CVI from Example 3, x be an instance of f , and ϕ be a p -isomorphism of f onto CVI. Then the function

$$E_f(x, w) = \phi^{-1}(E_{\text{CVI}}(\phi(x), w))$$

is an encryption function for f .

Because E_{CVI} , ϕ , and ϕ^{-1} run in polynomial time, take yes-instances to yes-instances, and take no-instances to no-instances, E_f does as well; thus E_f satisfies conditions 1 and 2 of Definition 1. If $x' \in \text{Range}(E_f)$, then $\phi(x') \in \text{Range}(E_{\text{CVI}})$; each instance in the preimage of $\phi(x')$ under E_{CVI} is of the form $\phi(x)$ for a *unique* instance x of f , because ϕ is a bijection. Thus $|\{x: \exists w: E_f(x, w) = x'\}| = |\{\phi(x): \exists w: E_{\text{CVI}}(\phi(x), w) = \phi(x')\}|$ is not $O(p(|x'|))$ for any polynomial p . (Actually, we see immediately that it is not $O(p(|\phi(x')|))$ for any polynomial p , but this is equivalent, because ϕ can cause only polynomial growth or shrinkage in the size of both yes- and no-instances [M].) This means that E_f satisfies condition 3. The proof that it satisfies condition 4 is almost identical. ■

The following theorem goes as far as we can go by combining Lemmas 2 and 3 with the results of [BH]. No definitive statement can be made about which NP-complete problems are encryptable under Definition 1 without settling the question of whether they are all p -isomorphic.

Theorem 1: All problems that are in the same p -isomorphism class as CNF-SAT are encryptable under Definition 1. No NP-complete problems are known to lie outside of this class.

Finally, we need to exhibit an encryption scheme for the discrete logarithm problem that satisfies Definition 1. First observe that it is possible to pose the problem in yes/no form. For fixed p and g , each instance is a pair $(x, [a, b])$, where the second argument is a subinterval of $[1, p - 1]$. All arithmetic is done in Z_p ; so if $a > b$ in Z , the elements of the subinterval are $a, a + 1, \dots, p - 1, 1, 2, \dots, b$. The answer to the instance $(x, [a, b])$ is “yes” if and only if there is an $e \in [a, b]$ such that $g^e \equiv x \pmod{p}$. Binary search is used to answer an instance of the standard discrete logarithm problem in $O(\log p)$ iterations of the yes/no version: First choose a random element e of $[1, p - 1]$, set $[a, b]$ to $[e, e + \frac{p-1}{2} - 1 \pmod{p - 1}]$, and set $[A, B]$ to $[1, p - 1]$. Then repeat these steps until the discrete logarithm of x is in hand: Submit the instance $(x, [a, b])$ to the yes/no version of the algorithm. If the answer is yes, then set $[A, B]$ to $[a, b]$ and set $[a, b]$ to a random subinterval of this new $[A, B]$ of size $\frac{|A-B|+1}{2}$. If the answer is “no”, then leave $[A, B]$ unchanged and set $[a, b]$ to its complement in $[A, B]$. The logarithm of x results from an affirmative answer to an instance of the decision problem in which $a = b$. In order to encrypt an instance of the standard version, choose a random c as in Example 1, go through the binary search with $g^c x \pmod{p}$ as the first argument and both endpoints of every interval translated by $c \pmod{p - 1}$, and subtract c from the final answer.

The results we get with Definition 1 are unsatisfactory. It is possible for an encryption scheme to meet the requirements and still be vulnerable to the criticism that it doesn’t really hide anything. This is the case in Example 3, where the possible preimages of an encrypted CVI instance are numerous, and they are *syntactically* different, but they have a lot of structure in common.

The encryption schemes we get for other NP-complete problems by applying Theorem 1 are just as weak in this respect, because they come from Example 3 via isomorphisms. In order to get a meaningful definition of encryptability based on the size of the preimages of encrypted instances, we'd have to make precise when we call two instances different enough to count them separately.

Another important shortcoming of this approach is that it gives no hint of how to prove negative results. Our intuition is that many problems of importance in cryptography, e.g. integer factoring, are *not* encryptable and that the right definition would enable us to prove this.

4. Directions of Current and Future Work

The failure of Definition 1 can be restated as follows: we have not said what the secret is. Exactly *what* about the cleartext instance x cannot be inferred from the encrypted instance x' without knowledge of the key k ? In Definition 2, we address this question and ignore completely the size of the preimage of x' , which was the focus of Definition 1.

Definition 2: Let f be a decision problem. We say that f is encryptable if there are two functions E_1 and E_2 and a set K of keys such that

$$1) E_i : \text{Dom}(f) \times K \rightarrow \text{Dom}(f), \quad i = 1, 2,$$

2) Both E_1 and E_2 are computable in polynomial time, and

3) $E_1(x, k)$ is a yes-instance of f if and only if x is a yes-instance, and $E_2(x, k)$ is a yes-instance if and only if x is a no-instance.

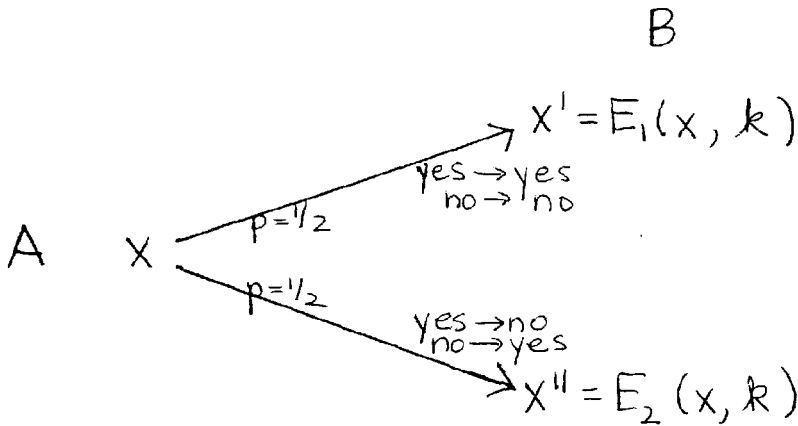


Figure 2: f is encryptable under Definition 2

So A is trying to hide the answer, $f(x)$. She chooses a key and, with probability $\frac{1}{2}$, uses the answer-preserving transformation E_1 , with probability $\frac{1}{2}$ the answer-reversing transformation E_2 . B tells her “yes” or “no” and has only a 50-50 chance of guessing which is the answer to her original instance.

The first thing to observe about Definition 2 is that it is unlikely to be satisfied by a problem that’s NP-complete: the function E_2 would be a polynomial-time reduction from f to its complement and thus could only exist if $\text{NP} = \text{Co-NP}$. However, there is a natural example of a decision problem for which such a pair E_1, E_2 can be found:

Example 4: Let $f(n)$, $n \in \mathbb{N}^+$, be “yes” if and only if n has an odd number of distinct prime factors. The key-space K consists of small sets of primes. A’s algorithm for E_1 is to pick an even number of primes p_1, \dots, p_{2t} and compute $E_1(x, \{p_1, \dots, p_{2t}\}) = np_1 \cdots p_{2t}$; to reverse the answer in E_2 , she does the same thing using an odd number of primes.

This f belongs to $\text{NP} \cap \text{Co-NP}$ and leads us to the

Open Question: Is every problem in $\text{NP} \cap \text{Co-NP}$ encryptable under Definition 2?

Finally, it would be very instructive to find some plausible definition under which we could prove that integer factoring is not encryptable. It is possible that some good would come of an attempt to generalize Definition 2 so that it applied to a broader class of f 's than just decision problems.

5. Acknowledgements

I would like to thank my advisor Andy Yao for many helpful discussions of these ideas. I also got a lot of useful feedback from friends at Stanford and at AT&T Bell Labs; special mention is due Eli Upfal for his suggestion that I look at Definition 2.

6. References

- [BH] Berman, Len and Juris Hartmanis. "On Isomorphisms and Density of NP- and other Complete Sets", *SIAM J. on Comput.*, 6 (2), 1977, 305-322.
- [CF] Coppersmith, Don and Joan Feigenbaum. "Finite Graphs with Two Inequivalent Factorizations Under the Composition Operator", IBM Research Report RC11149, 1985, submitted to *Journal of Combinatorial Theory, Series B*.
- [FS] Feigenbaum, Joan and Alejandro A. Schäffer, "Recognizing Composite Graphs is Equivalent to Testing Graph Isomorphism", to appear in *SIAM J. on Comput.*
- [M] Mahaney, Stephen R. "On the Number of p -isomorphism Classes of NP-complete Sets", Proc. 22nd Annual IEEE Symposium on the Foundations of Computer Science, 22, 1981, 271-278.
- [P] Plaisted, David. "Some Polynomial and Integer Divisibility Problems are NP-hard", Proc. 17th Annual IEEE Symposium on the Foundations of Computer Science, 17, 1976, 264-267.

DIVERGENCE BOUNDS ON KEY EQUIVOCATION AND ERROR PROBABILITY IN CRYPTANALYSIS

Johan van Tilburg and Dick E. Boeke

Delft University of Technology

Department of Electrical Engineering

Information Theory Group

P.O. Box 5031, 2600 GA Delft, The Netherlands

0. Abstract

A general method, based on the f -divergence (Csiszar) is presented to obtain divergence bounds on error probability and key equivocation. The method presented here is applicable for discrete data as well as for continuous data. As a special case of the f -divergence it is shown that the upper bound on key equivocation derived by Blom is of the Bhattacharyya type. For a pure cipher model using a discrete memoryless message source a recursive formula is derived for the error probability. A generalization of the β -unicity distance is given, from which it is shown why the key equivocation is a poor measure of theoretical security in many cases, and why lower bounds on error probability must be considered instead of upper bounds. Finally the concept of unicity distance is generalized in terms of the error probability and is called the Pe-Security Distance.

1. Introduction

Cipher systems have given birth to the possibility of sending secret messages via public insecure channels. The secrecy of the messages depends highly on the strength of the cipher system used. When evaluating the theoretical strength of cipher systems, it is assumed that the cryptanalyst behaves rationally, that he or she knows the set of transformations, the statistics of the message and the key source. The cryptanalyst tries to estimate the message used and/or the key from the intercepted cryptogram. Shannon [1] used a probabilistic model for the theoretical analysis of secrecy systems. This model has been refined recently by Jürgensen and Matthews [2].

In Shannon's paper it is pointed out that if the cryptanalyst intercepts a cryptogram, he is able to calculate the a posteriori probabilities of the various possible messages and keys which might have produced this cryptogram. This set of a posteriori probabilities describes how the cryptanalists knowledge of the message and key gradually becomes more precise as more enciphered text is intercepted. Shannon used as a measure of theoretical strength the equivocation which deals with a simplified description of the set of a posteriori probabilities. Then zero equivocation means that one key or message has a probability of one, and all others zero, corresponding to complete knowledge of the original key or message. Shannon also noticed that calculating the equivocation for the simplest type of cipher and language structure induces formulas which are nearly useless. His observation that the complexity of the problem suggests a method of approach, since sufficiently complicated problems can frequently be solved statistically, leads to the introduction of the famous "random cipher". Hellman [3] has shown that the random cipher actually defines a lower bound on the existence of good ciphers.

Blom [4] followed another way, by deriving an exponentially tight upper bound on the key equivocation for a simple substitution cipher (SSC) which is computationally more tractable. In Blom [5] an upper bound on the key equivocation for pure ciphers is given which exhibits the same structure as the bound in [4]. Later on, Dunham [6] derived bounds on the key appearance equivocation for an SSC and used the results of Blom [4] for bounding the message equivocation. Sgarro's paper [7] is based on an approach in coding theory, where one estimates error probabilities with respect to optimal coding problems. Sgarro made use of Kullback-Leibler divergence and composition classes to bound the error probability. His main results are asymptotic and contain the same relevant parameters as obtained by Blom [4] and Dunham [6].

2. Bounds with f-divergence

Typical in the approach is the use of information measures. For example, using Shannon's information measure leads to easy manipulation in a natural and intuitive way between different probability distributions (pd's). But still the underlying relevant parameter is the error probability (P_e). By bounding P_e with information measures, a region is determined in which the actual P_e can be found. The uncertainty in the value of P_e is resolved only in limiting cases where the bounds are tight. An excellent and straightforward use of this approach is given

by Lu [8], who uses Shannon's information measure to obtain the desired relation and applies the Fano inequality to lower bound P_e .

In the paper we consider the encipher system as a black box. Suppose we know the pd of the input (message) and the pd of the output (cryptogram). Transform the pd of the output to the input under a known key and compare the two pd's by means of P_e . Repeat this for all keys and select that key for which P_e is minimal. If ties occur then force a decision according to an arbitrary rule. Whereas determining P_e in a direct manner is quite involved, a much more natural way is to make use of the concept of distance measures since P_e is actually a distance measure itself.

The study of bounds on P_e has been of particular interest in the field of pattern recognition related to feature selection. Several distance measures have been used to obtain bounds on P_e , like Kolmogorov's variational distance, the Bhattacharyya distance, the J-divergence, the (generalized) Bayesian distance as well as many others. Much effort has been put into generalizing these measures from two classes with equal a priori probabilities to classes with non-equal a priori probabilities and from there to m classes, with $m \geq 2$. The comparison of the various bounds on P_e has also received much attention. More details can be found in Kanal [9] and in Chen [10].

A generalized approach can be given by using the f-divergence, as defined by Csiszar [11]. In this paper we shall use a slight modification, which we shall call the normalized average f-divergence. This divergence measure is directly related to P_e by its very definition, and it is therefore convenient for manipulating in this theoretical context. We shall use a definition which is sufficient for this paper. More details can be found in Boeke and van Tilburg [12] and in [13][14].

Before continuing, a short note about the notation. As far as possible the notation is in agreement with that of Blom in [4][5], with the exception that the logarithms involved are taken to base 2. Throughout this paper we shall use the convention that capital letters denote random variables, boldface letters denote sequences, capital script letters are reserved for sets and lower case letters represent the elements in a set.

Let \mathcal{S} denote an arbitrary (finite) set with cardinal number $|\mathcal{S}|$. \mathcal{S}^L is the class of all sequences \mathbf{s} of length L . A sequence (concatenation of symbols) \mathbf{s} of length L of elements s (not necessarily different) in \mathcal{S} is indicated by \mathbf{s}^L .

The cipher model is a set of uniquely reversible transformations of

$T = \{t_j\}_{j=1}^J$ of a set of possible messages $M = \{m_n\}_{n=1}^N$ into a set of cryptograms $E = \{e_n\}_{n=1}^N$, the transformations having associated probabilities $P = \{p_j\}_{j=1}^J$. J is the cardinal number of the set of keys $K = \{k_j\}_{j=1}^J$.

Definition 2.1: The normalized average f -divergence (for short: f -divergence) for isonorm-functions $f(x)$ is given by:

$$\bar{D}_f = \bar{D}_f(1;2) = \frac{f_\infty - \bar{D}_f(1;2)}{2 \cdot f_\infty - f_1},$$

where f is a convex function satisfying:

$$f_\infty = \lim_{x \rightarrow \infty} \frac{f(x)}{x}, \quad f_0 = \lim_{x \rightarrow 0} f(x),$$

$$f_0 = f_\infty \quad (\text{isonorm restriction}),$$

$$f_1 = f(1)$$

and

$$\bar{D}_f(1;2) = E_{E^L} \left[f \left(\frac{P_{K/E^L}(k_1/e^L)}{P_{K/E^L}(k_2/e^L)} \right) \cdot P_{K/E^L}(k_2/e^L) \right]$$

is the average f -divergence for discrimination of key k_1 against k_2 . By E_{E^L} we mean the expectation operator. \square

If we define $f_*(x) = x \cdot f(\frac{1-x}{x})$ and $u^L = u(e^L) = P_{K/E^L}(k_2/e^L)$, then it follows that:

$$f(u^L) = \frac{1}{2} \frac{f_*(0) - f_*(u^L)}{f_*(0) - f_*(\frac{1}{2})},$$

and hence $\bar{D}_f = E_{E^L} [f(u^L)]$.

Note that $Pe = Pe(K/E^L) = 1 - E_{E^L} [\max(u^L, 1-u^L)] = E_{E^L} [\min(u^L, 1-u^L)]$, which shows that the f -divergence includes the error probability as a special case for $f(x) = Pe(x) = \min(x, 1-x)$.

Definition 2.2: The Bhattacharyya distance is given by

$$B = -\log c,$$

where

$$c = c(E^L/K) = \sum_{e^L \in E^L} \sqrt{p_{E^L/K}(e^L/k_1) \cdot p_{E^L/K}(e^L/k_2)}$$

is the Bhattacharyya coefficient. \square

If we take the a priori probabilities of the keys into account, we obtain the next definition.

Definition 2.3. The average Bhattacharyya distance is given by

$$\bar{B} = -\log \bar{\rho},$$

where

$$\begin{aligned}\bar{\rho} &= \bar{\rho}(K/E^L) = E_{E^L} [\sqrt{P_{K/E^L}(k_1/e^L) \cdot P_{K/E^L}(k_2/e^L)}] \\ &= \sum_{e^L \in E^L} \sqrt{P_{KE^L}(k_1, e^L) \cdot P_{KE^L}(k_2, e^L)}\end{aligned}$$

is the average Bhattacharyya coefficient. \square

If the keys are equiprobable it follows that $\bar{\rho}(K/E^L) = \frac{1}{2} \rho(E^L/K)$. If we set $f(x) = -x^{1-a}$, we find that $f(x) = x^a \cdot (1-x)^{1-a}$. Then the f -divergence becomes

$$\bar{D}_f = E_{E^L} [f(u^L)] = E_{E^L} [(u^L)^a \cdot (1-u^L)^{1-a}],$$

which is the Chernoff distance $C_a(K/E^L)$. For $a=\frac{1}{2}$ we have

$$\bar{D}_f = E_{E^L} [\sqrt{u^L \cdot (1-u^L)}] = E_{E^L} [\sqrt{P_{K/E^L}(k_1/e^L) \cdot P_{K/E^L}(k_2/e^L)}],$$

which shows that the average Bhattacharyya coefficient is a special case of this f -divergence.

Similarly we find for $f(x) = |1-x|^{1/r}$:

$$\bar{D}_f = \frac{1}{2} - \frac{1}{2} \cdot \bar{M}_r^r,$$

where

$$\bar{M}_r = [E_{E^L} \{ \left| P_{K/E^L}(k_1/e^L)^{1/r} - P_{K/E^L}(k_2/e^L)^{1/r} \right|^r \}]^{1/r}$$

is the generalized Matusita distance.

For $r=1$ we have Kolmogorov's variational distance and for $r=2$ the usual Matusita distance.

In the next theorem a class of upper and lower bounds on the f -divergence is considered in terms of P_e . A sufficient condition for the validity of the theorem is to restrict the f -divergence to symmetric functions, i.e. $f(1-u)=f(u)$.

Theorem 2.1. A class of upper and lower bounds induced by Pe on the symmetric f -divergence is

$$Pe \leq \bar{D}_f \leq f(Pe).$$

Proof: First observe that $f(x)$ is a normalized concave function on $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$ resp., such that $f(x) \geq \min(x, 1-x)$ for $x \in [0, 1]$ with equality at least for $x \in \{0, \frac{1}{2}, 1\}$:

i) since $\min(u^L, 1-u^L) \leq f(u^L)$ it follows that

$$\bar{D}_f = E_{E^L}[f(u^L)] \geq E_{E^L}[\min(u^L, 1-u^L)] = Pe,$$

ii) $\bar{D}_f = E_{E^L}[f(u^L)] = E_{E^L}[f(\min(u^L, 1-u^L))] \leq f(E_{E^L}[\min(u^L, 1-u^L)])$
 $= f(Pe).$ \square

Remark. The theorem also gives bounds for normalized concave functions $f(x)$ which do not satisfy the f -divergence. Moreover the symmetric restriction is not used in the proof of the lower bound. So for the Chernoff bound it holds that $Pe \leq C_a(K/E^L)$.

In fact, the lower bound stated is a direct upper bound on $Pe(K/E^L)$. The upper bound in this theorem sometimes cannot be rewritten (explicitly) as a bound on $Pe(K/E^L)$. This can be a disadvantage if we are interested in bounds on $Pe(K/E^L)$. However, the lower bound on $Pe(K/E^L)$ can then be computed numerically or indirectly via the upper bound.

Example 2.1. Bounds for the average Bhattacharyya coefficient. Because then $f(x) = \sqrt{x \cdot (1-x)}$, we obtain

$$Pe \leq \bar{\rho}(K/E^L) \leq \sqrt{Pe \cdot (1-Pe)}$$

and

$$\frac{1}{2} \cdot (1 - \sqrt{1 - 4 \bar{\rho}^2}) \leq Pe(K/E^L) \leq \bar{\rho}.$$
 \square

Example 2.2. Bounds for the key equivocation. Because then $f(x) = \frac{1}{2} \cdot h(x) = \frac{1}{2} \cdot [-x \cdot \log x - (1-x) \cdot \log(1-x)]$, we obtain

$$Pe \leq \frac{1}{2} \cdot H(K/E^L) \leq \frac{1}{2} \cdot h(Pe),$$

where

$$H(K/E^L) = E_{E^L}[h(u^L)]$$

is Shannon's key equivocation. \square

The next lemma can easily be verified. A proof can be found, e.g., in Ito [15]. The lemma is illustrated in figure 2.1.

Lemma 2.1. The relation between the bounds is given by

$$Pe(K/E^L) \leq \frac{1}{2} H(K/E^L) \leq \bar{\rho}(K/E^L).$$

□

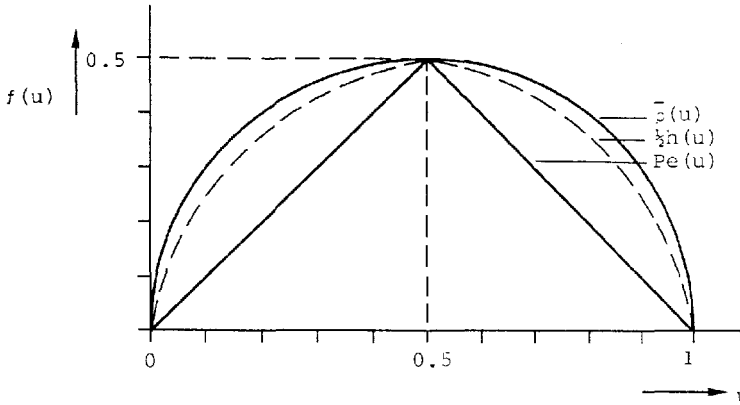


Figure 2.1. The basic functions which constitute the measures $\bar{\rho}$, $\frac{1}{2}H$ and Pe .

Next $\bar{\rho}$ will be determined for the general case, after which we shall return to the binary case. The model used is that of a pure cipher with the following assumptions.

- . The message and the key are stochastically independent.
- . The message source is discrete and memoryless.

$$\begin{pmatrix} M \\ Q_M \end{pmatrix} = \begin{pmatrix} 1 & 2 & \dots & N \\ q_1 & q_2 & \dots & q_N \end{pmatrix}.$$

- . T is the set of all unique invertible transformations t_j of M onto E , where the index j is the associated key.

$$T = \{t_j\}_{j=1}^J.$$

Note that $|K| = J$.

- . The cryptogram alphabet E is (not necessarily) identical to M .

$$\begin{pmatrix} E \\ Q_{t_j^{-1}(E)} \end{pmatrix} = \begin{pmatrix} 1 & 2 & \dots & N \\ q_{t_j^{-1}(1)} & q_{t_j^{-1}(2)} & \dots & q_{t_j^{-1}(N)} \end{pmatrix}.$$

Definition 2.4. Pure cipher (see Blom [16, theorem 3]).

A cipher is pure if and only if its set of enciphering transformations T is a coset (left or right) in G and

the keys are equiprobable. □

Remark. G is the multiplicative group of all invertible transformations of M onto M .

Following Blom [5], the set of enciphering transformations T forms a left coset in the group G . When the keys are equiprobable, it follows that the cipher is pure.

As T is a left coset, we may define

$$T = \{gr_j\}_{j=1}^J,$$

where $g \in G$ and $R = \{r_j\}_{j=1}^J$ is a subgroup in G .

Recalling Shannon [1], two secrecy systems R and S are similar, if there exists an invertible transformation A such that $R = A.S$. This means that enciphering with R is the same as enciphering with S followed by the transformation A . It is clear that similarity is an equivalence relation. The problem of finding bounds for a cipher using the set of transformations T is now transformed to a cipher using the set of transformations R , where R is a subgroup in G .

Theorem 2.2. The Bhattacharyya coefficient for the j^{th} transformation in a pure cipher model T using an N -ary discrete memoryless source with a priori probabilities q_n is given by

$$\rho_{1j} = \left(\sum_{n=1}^N \sqrt{q_n \cdot q_{r_j^{-1}(n)}} \right)^L,$$

where

$$\begin{aligned} r_j^{-1} &\in R, R \text{ is the group generating } T, \\ r_1 &\text{ is the identity element in } R. \end{aligned}$$

Proof. Because the keys are equiprobable and independent of the message source it follows for $L=1$ that

$$p_{EK}(e, k_1) = \frac{1}{J} \cdot p_M(r_{k_1}^{-1}(e)),$$

where

$$p_M(x) = \sum_{n=1}^N q_n \cdot \delta(n-x).$$

As stated we compare the pd of the message source with the inversely transformed encryption pd which depends on the transformation (key) used. By noting that k_1 is associated with the identity transformation, $\bar{0}$ becomes

$$\bar{\rho}_{1j} = \sum_{e \in E} \sqrt{p_{EK}(e, k_1) \cdot p_{EK}(e, k_j)}.$$

After substitution of $p_{EK}(e, k_i)$ one easily obtains

$$\bar{\rho}_{1j} = \frac{1}{J} \sum_{n=1}^N \sqrt{q_n \cdot q_{r_j^{-1}(n)}}.$$

$\bar{\rho}_{1j}$ for the extension of the cryptogram ($L > 1$) follows directly from the weak additivity of the Bhattacharyya distance B.

$$-L \log \rho = -\log \rho^L,$$

so that

$$\rho_{1j} = J \cdot \bar{\rho}_{1j} = \left(\sum_{n=1}^N \sqrt{q_n \cdot q_{r_j^{-1}(n)}} \right)^L.$$

□

For the binary case $\bar{\rho}_{1j}$ reduces to $\bar{\rho}_{12} = \bar{\rho} = \frac{1}{2}(\sqrt{4q_1q_2})^L$.

Substitution in lemma 2.1 and example 2.1 proves the next theorem.

Theorem 2.3. Bounds on the average probability of error (or probability of incorrect key identification) in a pure cipher model using a discrete memoryless source with a priori probabilities q_n are:

$$\frac{1}{2} \cdot (1 - \sqrt{(1 - (4q_1q_2)^L)}) \leq \text{Pe}(K/E^L) \leq \frac{1}{2} \cdot H(K/E^L) \leq \frac{1}{2} \cdot (\sqrt{4q_1q_2})^L.$$

□

The upper bound on the key equivocation is the same as obtained by Blom [4] using an SSC-model; however, at the same time we have a lower bound too. Moreover, for a different cipher model we only have to substitute the corresponding $\bar{\rho}$ in example 2.1, yielding the new upper and lower bounds. This illustrates the general structure of the bounds.

By a similar argument it can be shown that for the Chernoff bound it holds that

$$\text{Pe}(K/E^L) \leq C_a(K/E^L) = \frac{1}{2} \cdot (q_1^a \cdot q_2^{1-a} + q_2^a \cdot q_1^{1-a})^L,$$

where $0 \leq a \leq 1$.

This is a symmetric upper bound, which is minimal for $a = \frac{1}{2}$; that is, if it coincides with the Bhattacharyya bound. This shows that the Bhattacharyya bound is optimal in this context.

Thus far bounds on Pe have been considered. In the next theorem some recursive properties of Pe are stated.

Theorem 2.4. For the average probability of error (or probability of incorrect key identification) in a pure cipher model using a discrete memoryless source with a priori probabilities $p \geq q$ it holds that

- i) L is even: $Pe(K/E^{L+1}) = Pe(K/E^L) - \frac{1}{2}(p-q) \binom{L}{L/2} (\sqrt{pq})^L$
 with $Pe(K/E^0) = 0.5$
- ii) L is odd: $Pe(K/E^{L+1}) = Pe(K/E^L)$.

Proof

i) If L is even we have

$$\begin{aligned} Pe(K/E^{L+1}) &= \sum_{i=0}^{L+1} \left[\binom{L+1}{i} p^i q^{L+1-i} \cdot \min \left(\frac{p^i q^{L+1-i}}{p^i q^{L+1-i} + p^{L+1-i} q^i}, \frac{p^{L+1-i} q^i}{p^i q^{L+1-i} + p^{L+1-i} q^i} \right) \right] \\ &= \sum_{i=0}^{L/2} \left[\binom{L+1}{i} p^i q^{L+1-i} \right] = q \cdot \sum_{i=0}^{L/2} \left\{ \left[\binom{L}{i} + \binom{L}{i-1} \right] \cdot p^i q^{L-i} \right\} \\ &= q \cdot Pe(K/E^L) + \frac{1}{2} \binom{L}{L/2} (pq)^{L/2} + p \cdot \sum_{i=0}^{L/2-1} \left[\binom{L}{i} p^i q^{L-i} \right] \\ &= Pe(K/E^L) - \frac{1}{2}(p-q) \binom{L}{L/2} (pq)^{L/2}. \end{aligned}$$

ii) if L is odd we have

$$\begin{aligned} Pe(K/E^{L+1}) &= \sum_{i=0}^{L+1} \left[\binom{L+1}{i} p^i q^{L+1-i} \right] - \frac{1}{2} \binom{L+1}{\frac{L+1}{2}} (pq)^{L/2} \\ &= q \cdot \sum_{i=0}^{\frac{L+1}{2}} \left[\binom{L}{i} p^i q^{L-i} \right] + q \cdot \sum_{i=1}^{\frac{L+1}{2}} \left[\binom{L}{i-1} p^{i-1} q^{L-i+1} \right] - \frac{1}{2} \binom{L+1}{\frac{L+1}{2}} (pq)^{L/2} \\ &= q \cdot Pe(K/E^L) + \left(\frac{L}{\frac{L+1}{2}} \right) (pq)^{\frac{L+1}{2}} + p \cdot \sum_{i=1}^{\frac{L-1}{2}} \left[\binom{L}{i} p^i q^{L-i} \right] - \frac{1}{2} \binom{L+1}{\frac{L+1}{2}} (pq)^{L/2}. \end{aligned}$$

Since $\frac{1}{2} \cdot \binom{L+1}{\frac{L+1}{2}} = \binom{L}{\frac{L+1}{2}}$ it follows that $Pe(K/E^{L+1}) = Pe(K/E^L)$. □

An efficient algorithm can be obtained if theorem 2.4 is written in the following way.

$$\begin{aligned} P_e(0) &= 0.5, \\ A(0) &= p - 0.5, \\ B &= 4.p.(1-p), \end{aligned}$$

for L is even: $P_e(L+2) = P_e(L+1) = P_e(L) - A(L)$
 $A(L+2) = A(L) \cdot (1 - 1/(L+2)) \cdot B.$

Remark. Although the key equivocation is a simplified description of the set of a posteriori probabilities, it cannot be transformed into an effective algorithm of the above type for an SSC-model.

From the theorem we may conclude that the behaviour of P_e for small L is determined by $|p-q|$. This is in contrast to the longterm exponential behaviour which is characterized by $|\sqrt{p} - \sqrt{q}|$.

A discussion of the relations between the different bounds and the actual average error probability is deferred to section 4, where the variance of the bound is investigated too.

6. Bound extensions of the Bhattacharyya type

In section 2 we have considered bounds on P_e for cipher systems using binary sources. We now turn to the N -ary problem where a general bound in terms of the f -divergence can be given. Generally speaking this generalized bound becomes less tight for increasing N . However, some particular functions allow better bounds. For this reason this section is completely devoted to the extension of the Bhattacharyya bound. Furthermore there exists a general class of distance measures (for instance the general mean distance [14]) which are inherently based on the N -ary problem. First, a general bound of the Bhattacharyya type is derived on $P_e(K/E^L)$ as well as on $H(K/E^L)$, after which the bound is restricted to the pure cipher model. Finally, the pure cipher bound is applied in the case of a discrete memoryless source.

A direct extension of the Bhattacharyya bound can be found by making use of the following theorem:

Theorem 3.1. The upper bounds on the probability of error in a cipher model using an N -ary source are given by

$$P_e(K/E^L) \leq \frac{1}{2} \cdot H(K/E^L) \leq \frac{1}{2} \cdot \log \left(\sum_{i=1}^J \sum_{j=1}^J \bar{c}_{ij} \right),$$

where

$$\bar{c}_{ij} = E_{E^L} \left[\sqrt{P_{K/E^L}(k_i/e^L) \cdot P_{K/E^L}(k_j/e^L)} \right].$$

Proof. i) Kovalevski [17] has proved that

$$\log m + m \cdot (m+1) \cdot \log\left(\frac{m+1}{m}\right) \cdot (Pe(K/E^L) - \frac{m-1}{m}) \leq H(K/E^L), \text{ with } \frac{m-1}{m} \leq Pe \leq \frac{m}{m+1}.$$

For $m=1$ or $0 \leq Pe \leq \frac{1}{2}$ this bound reduces to $2 \cdot Pe(K/E^L) \leq H(K/E^L)$. This holds for $Pe > \frac{1}{2}$ too, with the implication that the implicit expression is tighter.

$$\begin{aligned} \text{ii) } H(K/E^L) &= E_{E^L} \left[H(K/e^L) \right] = E_{E^L} \left[- \sum_{i=1}^J P_{K/E^L}(k_i/e^L) \cdot \log P_{K/E^L}(k_i/e^L) \right] \\ &\leq E_{E^L} \left[\log \left(\sum_{i=1}^J P_{K/E^L}(k_i/e^L) \right)^2 \right] \\ &= E_{E^L} \left[\log \sum_{i=1}^J \sum_{j=1}^J \sqrt{P_{K/E^L}(k_i/e^L) \cdot P_{K/E^L}(k_j/e^L)} \right] \\ &\leq \log \sum_{i=1}^J \sum_{j=1}^J \bar{\rho}_{ij}. \end{aligned}$$

This result has implicitly been proved by Blom [4].

Combining (i) and (ii) yields the theorem. □

For the pure cipher model we have

$$\begin{aligned} \cdot \text{ independence of keys used: } & \sum_{j=1}^J \bar{\rho}_{ij} = \sum_{j=1}^J \bar{\rho}_{kj}, \\ \cdot \text{ equiprobable keys } & : \bar{\rho}_{ij} = \frac{1}{J} \quad \text{for } i=j, \end{aligned}$$

which implies the next corollary.

Corollary 3.1. For the upper bounds on the probability of error in a pure cipher model using an N-ary source, we have

$$Pe(K/E^L) \leq \frac{1}{2} H(K/E^L) \leq \frac{1}{2} \log \left(1 + \sum_{i=2}^J \rho_{1j} \right),$$

where

$$\rho_{1j} = J \cdot \bar{\rho}_{1j} \text{ and } J = |K|.$$
□

The next corollary ensues from substituting ρ_{1j} (Theorem 2.2) in corollary 3.1. Because the summation of ρ_{1j} is taken over all transformations in the group \mathcal{R} it makes no difference if we write $r_j(n)$ instead of $r_j^{-1}(n)$.

Corollary 3.2. For the upper bounds on the probability of error in a pure cipher model T using an N -ary discrete memoryless source, we have

$$Pe(K/E^L) \leq \frac{1}{2} H(K/E^L) \leq \frac{1}{2} \log \left[1 + \sum_{j=1}^J \left(\sum_{n=1}^N \sqrt{q_n \cdot q_{r_j(n)}} \right)^L \right],$$

here

$j \in R$, R is the group generating T and r_1 is the identity element. \square

The upper bound on the key equivocation is the same as obtained by Blom [5]. However, the proof is simplified considerably and the general structure of the bound becomes clear.

A lower bound can be found by using the natural multiplicative extension of the Bhattacharyya coefficient. A general (non-trivial) upper bound for this extension does not exist (van Tilburg [18]).

Theorem 3.2. A lower bound on the probability of error in a cipher model using an N -ary source is given by

$$1 - Pe(K/E^L) \cdot Pe(K/E^L)^{J-1} \geq (J-1)^{J-1} \cdot \bar{\rho}_J^J,$$

$$i) \quad Pe(K/E^L) \geq (J-1) \cdot \bar{\rho}_J^{\frac{J}{J-1}},$$

where $\bar{\rho}_J = E_{E^L} \left[\prod_{j=1}^J P_{K/E^L}(k_j/e^L)^{1/J} \right]$ is the multiplicative extension of the average Bhattacharyya coefficient.

Proof. i) Define $x = x(e^L) = \max_j \left[P_{K/E^L}(k_j/e^L) \right]$. Then

$$\begin{aligned} \bar{\rho}_J &= E_{E^L} \left[\prod_{j=1}^J P_{K/E^L}(k_j/e^L)^{1/J} \right] \leq E_{E^L} \left[x^{1/J} \cdot \left(\frac{1-x}{J-1} \right)^{\frac{J-1}{J}} \right] \\ &\leq (E_{E^L}[x])^{1/J} \cdot (E_{E^L} \left[\frac{1-x}{J-1} \right])^{\frac{J-1}{J}} = (1 - Pe(K/E^L))^{1/J} \cdot \left(\frac{Pe(K/E^L)}{J-1} \right)^{\frac{J-1}{J}} \end{aligned}$$

$$\text{or } (J-1)^{J-1} \cdot \bar{\rho}_J^J \leq (1 - Pe(K/E^L)) \cdot Pe(K/E^L)^{J-1}.$$

i) Simplifying the inequality in (i) by making use of

$$(1 - Pe) \cdot Pe^{J-1} \leq Pe^{J-1}$$

implies

$$(J-1) \cdot \bar{\rho}_J^{\frac{J}{J-1}} \leq Pe(K/E^L). \quad \square$$

The proof of the next lemma is similar to that of theorem 2.2. and is therefore omitted.

Lemma 3.1. The multiplicative extension of the Bhattacharyya coefficient in a pure cipher model T using an N -ary discrete memoryless source with a priori probabilities q_n is given by

$$\rho_J = \left[\sum_{n=1}^N \prod_{j=1}^J q_{r_j(n)}^{1/J} \right]^L,$$

where

$$r_j \in R, R \text{ is the group generating } T \text{ and } \rho_J = J \cdot \bar{\rho}_J.$$

Substituting lemma 3.1 in theorem 3.1 yields corollary 3.3.

Corollary 3.3. A lower bound on the probability of error in a pure cipher model T using an N -ary discrete memoryless source with a priori probabilities q_n is given by

$$\frac{J-1}{J} \cdot \left[\sum_{n=1}^N \prod_{j=1}^J q_{r_j(n)}^{1/J} \right]^{\frac{J}{J-1} \cdot L} \leq \text{Pe}(K/E^L),$$

where

$$r_j \in R \text{ and } R \text{ is the group generating } T.$$

For large-sized key spaces we have the tight approximation

$$\left[\sum_{n=1}^N \prod_{j=1}^J q_{r_j(n)}^{1/J} \right]^L \leq \text{Pe}(K/E^L).$$

4. The Pe-security distance

The f -divergence is defined in a probabilistic environment and therefore easily fits into the probabilistic model of cryptosystems proposed by Jürgensen and Matthews [2]. In their paper (section 6) they have defined the β -UD as $\min_L \{L | H(K/E^L) \leq \beta\}$.

They also propose the (α, β) -security distance: a system is said to be (α, β) -secure at L if $\Pr\{H(K/E^L) \leq \beta\} \leq \alpha$. In the present section the β -UD is related to Pe and is not restricted to the key equivocation only. To avoid confusion we refer to this generalized β -UD as the γ -UD. When discussing the results of the SSC-model it is observed that Pe is a natural (theoretical) security measure. By noting this, Pe is derived

for a random cipher (RC-model). As a result it is found that at unicity distance P_e highly depends on the size of the key space. Hence we conclude that linking the UD to P_e leads to a better and more adequate explanation of the unicity distance. Finally, the concept of JD is generalized in terms of P_e and is called the P_e -security distance (P_e -SD). This security distance can be considered as a special case of the γ -UD and includes the original UD found in an RC-model too. Moreover, it becomes clear that lower bounds are needed to approximate the P_e -SD. For the key equivocation this means that one must make use of the Fano-inequality, because the key equivocation itself defines an upper bound.

In this section our main concern is the binary case. For this reason and to avoid unnecessary notational problems the γ -UD is mainly described for the binary case.

Definition 4.1. The generalized β -unicity distance or, for short, the γ -UD is defined as

$$L(\gamma) = \min_L \{L \in \mathbb{R}^+ \mid E_L[g(u^L)] \leq \gamma\},$$

where

$$u^L = u(e^L) = P_{K/E^L}(k_2/e^L),$$

and $g(\cdot)$ is a normalized function such that $g(x) \geq \min(x, 1-x)$ for $x \in [0, 1]$ with equality at least for $x \in \{0, \frac{1}{2}, 1\}$.

□

Observe that for $g(u^L) = f(u^L)$ we have the γ -UD for the f -divergence, whereas if $g(u^L) = \frac{1}{2}h(u^L)$ the γ -UD for the normalized key equivocation is obtained.

The γ -UD not only depends on the measure used, but it depends on the model used (including the source) too. This is illustrated by the following examples:

Example 4.1. For the γ -UD using the key equivocation we have

$$L(\gamma) = \min_L \{L \in \mathbb{R}^+ \mid \frac{1}{2}H(K/E^L) \leq \gamma\}.$$

If the key and message sources are independent we find

$$L(\gamma) = \min_L \{L \in \mathbb{R}^+ \mid H(E^L) - H(M^L) \geq H(K) - 2\gamma\}.$$

If in addition the message source is memoryless this becomes

$$L(\gamma) = \min_L \{L \in \mathbb{R}^+ | L \geq L(0) \cdot (1 - \frac{2\gamma}{H(K)})\},$$

with

$$L(0) = \frac{H(K)}{H(E) - H(M)}.$$

Note that $H(K/E^L)$ is convex in the sense that $H(K/E^L) - H(K/E^{L+1}) \geq H(K/E^{L+1}) - H(K/E^{L+2})$ and $H(E) - H(M) = H(K/E^0) - H(K/E^1)$, so that $L(0)$ can be found as the point of intersection of the straight line through $H(K/E^0)$ and $H(K/E^1)$ with the L -axis. This line defines a lower bound on the key equivocation. \square

Example 4.2. For a random cipher model we have

$$L(\gamma) = \min_L \{L \in \mathbb{R}^+ | L \geq L(0) \cdot (1-2\gamma)\},$$

with

$$L(0) = \frac{\log |K|}{\log |M| - H_L(M)}.$$

$L(0)$ is the original UD for the RC-model obtained by Shannon [1]. $H_L(M)$ denotes the entropy per symbol in a sequence of L message symbols, i.e. $H_L(M) = H(M^L)/L$. Note that the decrease of $L(\gamma)$ is linear; this is not necessarily so for other models. \square

Now we are able to state the next lemma which is a generalization of the second part of proposition 7.6 [2] with the assumptions made above. The proof is similar and is therefore omitted.

Lemma 4.1. If L_0 is the γ -unicity distance, then for $L > L_0$ we have

$$\Pr\{g(u^L) \leq \gamma\} \geq \frac{\text{Var}_{E^L}[g(u^L)]}{\gamma^2 + \text{Var}_{E^L}[g(u^L)]}.$$

\square

Let us consider an SSC-model using a binary memoryless message source with a priori probabilities $p=0.6$ and $q=0.4$. The upper and lower bounds on P_e (derived in section 2) are applied to this model and illustrated in figure 4.1, in which the exact value of P_e is given too. The figure shows that \bar{p} and $\frac{1}{2}H$ are loose upper bounds for small values of L ; even at UD they are still not tight. This is also demonstrated by the next example.

Example 4.3. (see also example 7.2 [2]).

Consider the SSC-model with $p=0.7$ and $L=7$ (RC-model: $UD=8.4$). For this model we have $\bar{\rho}(K/E^7) = 0.27$ and $\frac{1}{2}H(K/E^7) = 0.22$.

Although these values are not too high they are still much too optimistic since $Pe(K/E^7) = 0.126$. □

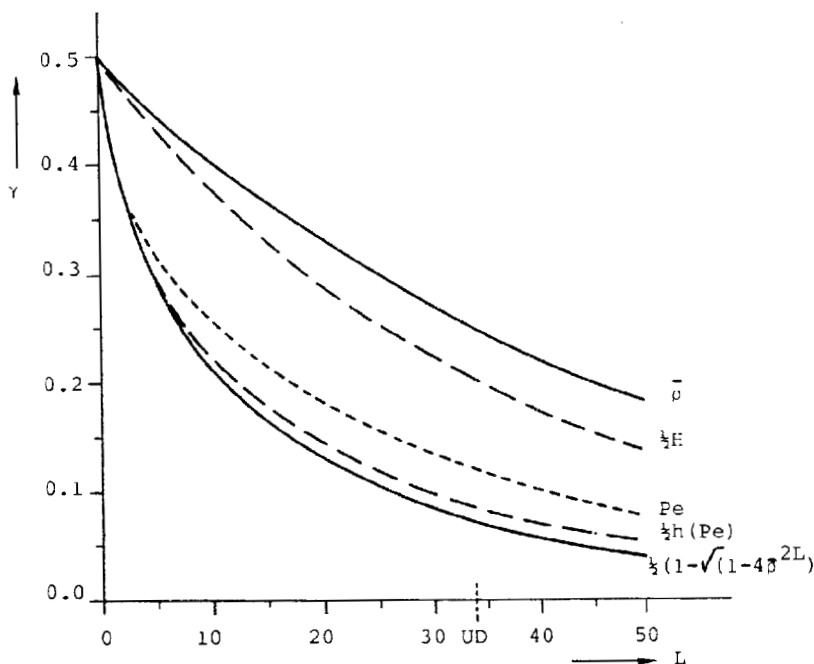


Figure 4.1. Bounds on the average probability of incorrect key identification Pe in a memoryless SSC-model with $p=0.6$.

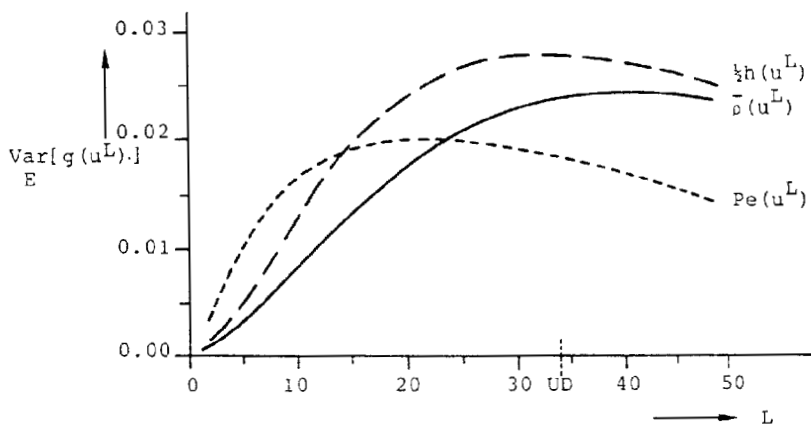


Figure 4.2. The variance of $\bar{\rho}(u^L)$, $\frac{1}{2}h(u^L)$ and $Pe(u^L)$ for a memoryless SSC-model with $p=0.6$.

In addition to all this, consider $L(\gamma)$ with γ constant for the different bounds. Now it becomes clear that the β -UD (and thus the (α, β) -SD, too) is a poor and positively biased estimator of L in $\text{Pe}(K/E^L) = \gamma$; this in contrast to the lower bounds which are negatively biased and tighter.

In figure 4.2. the variance of $g(u^L)$ is shown for the SSC-model. It is observed that the variance of $H(K/e^L)$ is maximal at UD, which is found for other values of p too. Moreover, the length of the cipher text at which the variance of $\text{Pe}(u^L)$ reaches its maximum is always less than the length obtained by $\bar{p}(u^L)$ and $h(u^L)$. This can be explained from the convex nature of $\text{Pe}^2(u^L)$; this in contrast to $\bar{p}^2(u^L)$, which is a concave function (see also figure 4.3). Besides this, for the normalized functions it holds that $\text{Var}[g(u^L)] \leq E[g(u^L)]$ since $\text{Var}[g(u^L)] \leq E[g^2(u^L)] \leq E[g(u^L)]$. This is illustrated by the next example.

Example 4.4. (see also p. 292 [3] and p. 343 [2]).

Suppose that after intercepting L enciphered symbols it holds that

$$n_k \begin{cases} 0 & 1 - 10^{-10} \\ 10^{20} & 10^{-10}, \end{cases} \quad \text{with probability}$$

in which n_k is the number of spurious key decipherments. Then $\bar{n}_k = 10^{10}$ and $\text{Var}(n_k) \approx 10^{30}$.

In the worst case $P_{K/E^L}(k_i/e^L) = P_{K/E^L}(k_j/e^L)$ for all k_i and k_j in K , so that the key space must satisfy $|K| = 10^{20} + 1$. For Pe we then obtain

$$\text{Pe}(K/E^L) \approx 10^{-10} \quad \text{and} \quad \text{Var}_{E^L}[\text{Pe}(u^L)] \approx 10^{-10}.$$

Since the real key space may be larger, say for example $|K| = 10^{100}$ it follows that

$$\text{Pe}(K/E^L) \approx 10^{-90} \quad \text{and} \quad \text{Var}_{E^L}[\text{Pe}(u^L)] \approx 10^{-170}.$$

So, the interpretation of \bar{n}_k (and $H(K/E^L)$ also) depends greatly on the size of the key space. For this reason it is necessary to utilize normalized functions. Moreover, the interpretation of the variance becomes more realistic too. □

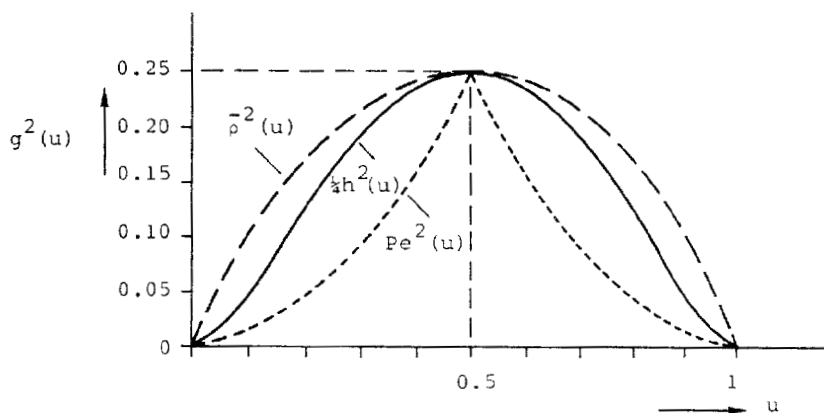


Figure 4.3. The variance weighting for $\bar{\rho}$, H and Pe .

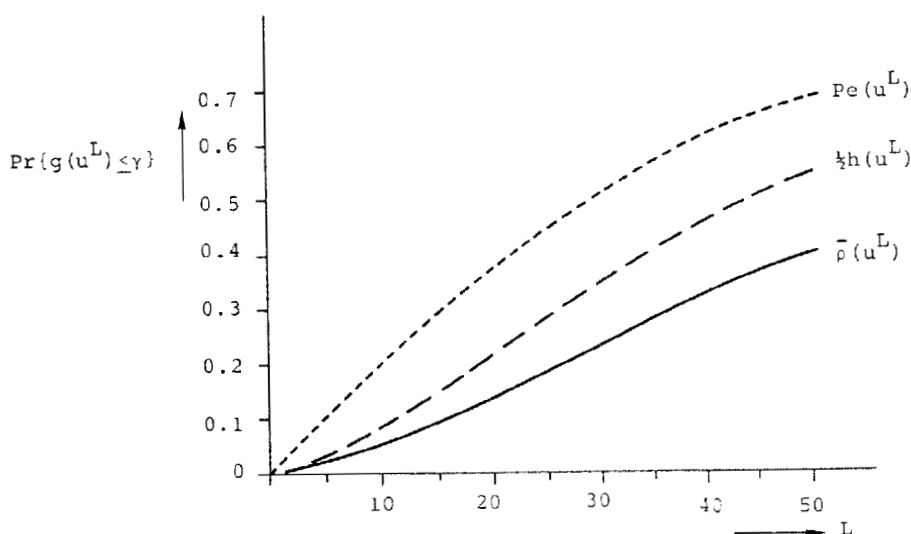


Figure 4.4. Lower bounds (Lemma 4.1) on $\Pr\{g(u^L) \leq \gamma\}$ for the different measures in a memoryless SSC-model with $p = 0.6$.

Figure 4.4. in combination with figure 4.1 tells us that $Pe(K/E^L)$ for a given γ is more reliable than the other measures. Finally it is observed that in the SSC-model $Pe(K/E^L)$ at $L=UD$ (UD in RC-model) is almost constant for different values of p and is approximately 0.12 even for $UD = 10^6$.

From the SSC-model it seems that Pe is a good and natural measure of theoretical security. For this reason we shall briefly pay attention to its behaviour in an RC-model.

The next theorem is a direct consequence of Hellman's definition of an RC-model and the expected number of spurious key decipherments [3].

Theorem 4.1. The average probability of error (or probability of incorrect key identification) in a random cipher model is given by

$$Pe_{RC}(K/E^L) = \frac{|K| - 1}{|K|} \cdot 2^{-L \cdot R} \approx 2^{-L \cdot R},$$

where

$$R = \log |M| \cdot \left(1 - \frac{H_L(M)}{\log |M|}\right).$$

Proof. There are $|K|$ different and independent keys so that

$Pe_{RC}(K/E^L) = \frac{\bar{n}_k}{|K|}$ in which \bar{n}_k is the average number of spurious key decipherments. According to Hellman [3, theorem 1] we have $\bar{n}_k = (|K| - 1) \cdot 2^{-L \cdot R}$ with $R = \log |M| - H_L(M)$. Substitution yields the theorem. If the key space is sufficiently large we have the nice approximation $2^{-L \cdot R}$. □

In a similar way the other theorems in [3] can be adapted in terms of Pe too.

Remark. It is important that the assumptions imposed by the RC-model be reasonable for the real secrecy system including the language used. For example, not only the uniformly distributed assumption must be considered but also the effective size of the key space which depends highly on the language used and on the length of the intercepted text. For large L the dependence may be negligible, but for small and moderate values one has to face the fact that some of the keys act similarly, i.e. key residue classes must be considered instead of the single keys. If a key residue class is detected with a small probability of error the remaining keys in this class are indistinguishable. At best one can choose a key according to an arbitrary rule. This introduces an extra error which depends on the size of the residue class. Note that data compression reduces this extra error. So when one's aim is to protect the key, data compression must be considered with care.

At unicity distance it holds that $H(K) = L \cdot R$. Now the next corollary follows immediately from theorem 4.1 and the corresponding remark.

Corollary 4.1. The average probability of error (or probability of incorrect key identification) in a random cipher model at unicity distance is given by

$$Pe_{RC}(K/E^{UD}) = \frac{|K|-1}{|K|} \approx \frac{1}{|K|}.$$

□

Note that $Pe_{RC}(K/E^{UD}) = 0.25$ for $|K| = 2$. For the SSC-model we have found that at $L=UD$ (UD in RC-model) $Pe_{SSC}(K/E^{UD}) \approx 0.12$, which was fairly constant even for a $UD=10^6$. This discrepancy is due to the fact that $Pe_{RC}(K/E^L)$ is an upper bound on $Pe_{SSC}(K/E^L)$ and is tight for $L \gg UD$.

Example 4.5. In an SSC-model using the English language for small and moderate values of L the effective number of keys is less than 26! This is caused by the fact that the average number of different letters that occur in messages of length L is less than 26. This is illustrated in table 4.1. At UD in an RC-model the average number of different letters per message is about 14. Therefore the average probability of error becomes

$$Pe_{RC}(K/E^{UD}) \approx \frac{12!}{26!} \approx 1.10^{-18}.$$

This means that on the average 1 key residue class to every 10^{18} key residue classes will be incorrectly identified from the effective number of keys induced by the ciphertext of UD length. The actual $Pe_{RC}(K/E^{UD})$ depends on the size of the key residue class too, which may be rather large. Nevertheless when we know the key residue class we know the message too. This explains why it is almost always possible to get a unique solution at UD.

□

As stated in corollary 4.1 the UD in an RC model defines a Pe which depends on the size of the key space (the larger the size of the key space, the smaller Pe). As a result the meaning of the UD for different sizes of the key space is also different, in the sense of Pe . Actually that is not what one prefers. It is desirable to have a UD for which the explanation is independent of the size of the key space. From the above arguments it seems that linking the UD to Pe leads to a better and more adequate explanation of the UD. For this reason we will generalize the concept of UD in terms of Pe and call the new distance the Pe -security distance (Pe -SD).

Message length L (Characters)	Average number of different letters per message
5	4.5
10	7.8
15	10.2
20	12.0
25	13.4
30	14.5
40	16.1
50	17.3
75	19.2
100	20.4
200	22.4
300	23.0
400	23.4
500	23.7
700	24.2
1000	24.6
1500	25.2

Table 4.1. The average number of different letters in L letters of English text. This table was adapted from Meyer and Matyas [19, table 12.3]

Definition 4.2. The Pe-security distance is defined by

$$L_m(\gamma) = \min_L \{L \in \mathbb{R}^+ \mid \text{Pe}_m(K/E^L) \leq \gamma\},$$

where

m is the actual cipher model, and

γ is a value of Pe. □

Remark. Depending on what one's object is (the key or the message), the Pe-security distance (for the N-ary case) can be based on $\text{Pe}_m(K/E^L)$ or on $\text{Pe}_m(M/E^L)$. From the definition it follows that the Pe-SD depends on the model "m" used (including the source) and the desirable value of Pe " γ ". The average performance of the Pe-SD is natural and clear.

Corollary 4.2. The Pe-security distance includes the original unicity distance in a random cipher model as a special case.

Proof. After substitution of

$$\text{Pe}_{RC}(K/E^L) = \frac{|K| - 1}{|K|} \cdot 2^{-L \cdot R} \quad \text{and} \quad \gamma = \frac{|K| - 1}{|K|^2},$$

with $R = \log |M| \cdot (1 - \frac{H_L(M)}{\log |M|})$, one easily obtains

$$\min_L \{L \in \mathbb{R}^+ | L \geq \frac{\log |K|}{R}\},$$

which is the original UD in an RC-model. □

For the SSC-model with redundancy R the Pe-SD characteristics are given in figure 4.5 for different values of γ . Note that Pe at UD is almost constant, in conformity with the predictions from the RC-model.

If determining $L_m(\gamma)$ in a direct manner is quite involved one can make use of the lower bounds given in the previous sections.

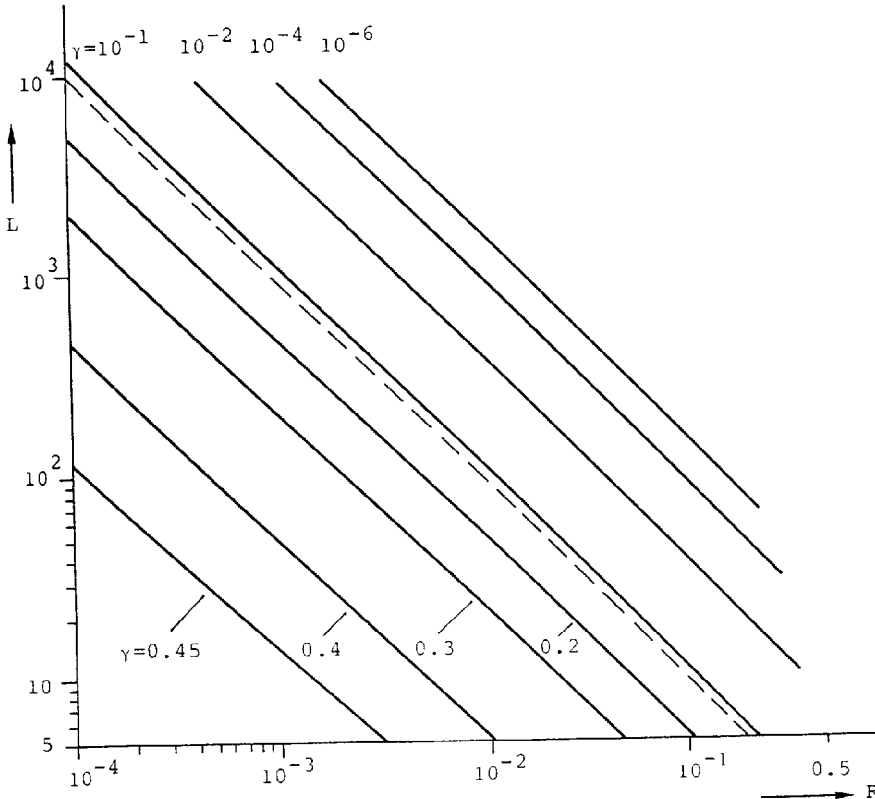


Figure 4.5. The Pe-SD characteristics for an SSC using a binary memoryless source with redundancy R . The dotted line represents Pe at UD.

Example 4.6. For a PC-model using a discrete memoryless source with a priori probabilities p and q we have for the Bhattacharyya coefficient

$$L_{PC}(\gamma) \geq \min_L \{L \in \mathbb{R}^+ \mid \frac{1}{2} \cdot (1 - \sqrt{1 - (4pq)^L}) \leq \gamma\},$$

from which it is easily found that

$$L_{PC}(\gamma) \geq \frac{\log [1 - (1-2\gamma)^2]}{\log(4pq)}.$$

□

The Pe-SD can be applied in the reverse direction too, i.e. for a given L the corresponding expected value of γ can be found. Using the same arguments lower bounds on Pe can be considered to determine γ .

Example 4.7. Again, consider an SSC using a discrete memoryless source with $p=0.7$ and $L=7$ [2, example 7.2]. Jürgensen and Matthews stated that this system is highly insecure even though $H(K/E^7) \approx 0.44$ is fairly large. Since $H(K/E^L)$ itself defines an upper bound on $Pe(K/E^L)$, one must make use of Fano's inequality $H(K/E^L) \leq H(Pe) + Pe \cdot \log(N-1)$. From $N=2$ and $H(K/E^7) \approx 0.44$ it is found that $\gamma \geq 0.09$. Therefore we may conclude that the system for the given source is indeed insecure.

□

It is illustrated by the example why the key equivocation itself, judged as measure of theoretical security, behaves poorly: it is an upper bound and usually only tight for large L . Although the key equivocation may be a poor measure of security in many cases, it certainly does not degrade the use of Shannon's information measure in cryptanalysis. The strength of this measure can be explained by the natural interpretation and accordingly by the convenient way of manipulating between different pd 's. For example, this has been demonstrated by Lu [8].

References

- [1] C.E. Shannon, Communication theory of secrecy systems, Bell Syst. Tech. J. 28, pp. 656-715, (1948).
- [2] H. Jürgensen and D.E. Matthews, Some results on the information theoretic analysis of cryptosystems, Proc. of CRYPTO'83, Santa Barbara, California, August 1983, pp. 303-356.
- [3] M.E. Hellman, An extension of the Shannon Theory Approach to Cryptography, IEEE Trans. Inform. Theory IT-23, pp. 289-294 (1977).

- [4] R. Blom, Bounds on Key Equivocation for Simple Substitution Ciphers, *IEEE Trans. Inform. Theory* IT-25, pp. 8-18 (1979).
- [5] R. Blom, An Upper Bound on the Key Equivocation for Pure Ciphers, *IEEE Trans. Inform. Theory* IT-30, pp. 82-84 (1984).
- [6] J.G. Dunham, Bounds on Message Equivocation for Simple Substitution Ciphers, *IEEE Trans. Inform. Theory* IT-26, pp. 522-527 (1980).
- [7] A. Sgarro, Error Probabilities for Simple Substitution Ciphers, *IEEE Trans. Inform. Theory* IT-29, pp. 190-198 (1983).
- [8] S.C. Lu, The Existence of Good Cryptosystems for Key Rates Greater than the Message Redundancy, *IEEE Trans. Inform. Theory* IT-25, pp. 475-477 (1979).
- [9] L. Kanal, Patterns in pattern recognition: 1968-1974, *IEEE Trans. Inform. Theory* IT-20, pp. 697-722 (1974).
- [10] C.H. Chen, Statistical pattern recognition, Hayden Book Co., Rochelle Park, New Jersey (1973).
- [11] I. Csizsar, Information-type measures of difference of probability distributions and indirect observations, *Stud. Sci. Math. Hungary.* 2, pp. 299-318 (1967).
- [12] D.E. Boekee and J. van Tilburg, Bounds on the Bayesian Error Probability using Concave Functions, to appear.
- [13] D.E. Boekee and J.C. Ruitenbeek, A Class of Lower Bounds on the Bayesian Probability of Error, *Information Sciences* 25, pp. 21-35, (1981).
- [14] D.E. Boekee and J.C.A. van der Lubbe, Some Aspects of Error Bounds in Feature Selection, *Pattern recognition*, Vol. 11, pp. 353-360 (1979).
- [15] T. Ito, Approximate Error Bounds in Pattern Recognition, *Machine Intelligence*, Vol. 7, pp. 369-376, Edinburgh Univ. Press (1972).
- [16] R. Blom, On Pure Ciphers, Internal. Rep. LiTH-ISY-I-0286, Linköping University, Sweden (1979).
- [17] V.A. Kovalevsky, On the Criteria for the Information Content of a System of Features, In: *Image Pattern Recognition*, pp. 67-90, (1980).
- [18] J. van Tilburg, Decisions and Selections based on the Bayesian Error Probability with Shannon Information, Certainty and f-divergence, Thesis, Delft Univ. of Techn. (1984, in Dutch).
- [19] C.H. Meyer and S.M. Matyas, *Cryptography: a new dimension in computer data security*, Wiley, NY (1982).

**A chosen text attack on the RSA cryptosystem
and some discrete logarithm schemes**

Y. Desmedt

Aangesteld Navorsers NFWO
Katholieke Universiteit Leuven
Laboratorium ESAT
B-3030 Heverlee, Belgium

A. M. Odlyzko

AT&T Bell Laboratories
Murray Hill, NJ 07974, USA

ABSTRACT

A new attack on the RSA cryptosystem is presented. This attack assumes less than previous chosen ciphertext attacks, since the cryptanalyst has to obtain the plaintext versions of some carefully chosen ciphertexts only once, and can then proceed to decrypt further ciphertexts without further recourse to the authorized user's decrypting facility. This attack is considerably more efficient than the best algorithms that are known for factoring the public modulus. The same idea can also be used to develop an attack on the three-pass system of transmitting information using exponentiation in a finite field.

1. Introduction

The RSA cryptosystem [13] is perhaps the most famous public key cryptosystem and, together with the Diffie-Hellman key exchange scheme [6], is one of the most important public key systems. It is often thought that breaking the RSA system is as hard as factoring the public modulus n used in the system, but this has never been proved. The attack by Simmons and Norris [14] involving repeated encryptions has been shown to be unlikely to succeed if the primes dividing the modulus n are chosen carefully [12]. On the other hand, it has been pointed out that there are ways to employ the RSA system that can be cryptanalyzed without factoring n . For example, Knuth's proposal to use a small encryption exponent (to speed up operation) was shown to be unsafe when the same message is being sent to several destinations simultaneously [1] (see also [4; pp. 57-58] and [7]).

Another attack on some particular ways to employ the RSA system is due to Davida [3] (see also [5]), and our result is similar to it and can be considered as a generalization of it. In that attack, suppose that n is the public modulus of user A and E the public encryption exponent. Suppose that the cryptanalyst intercepts the ciphertext $c \equiv m^E \pmod{n}$, and wishes to recover the plaintext m . He chooses a random integer x and forms

$$c' \equiv c x^E \pmod{n}.$$

If he can now get user A to decrypt c' (for example, if A uses the pair (E, n) for signatures and is willing to sign challenge messages, or else if A discards decrypted messages that appear meaningless, and the cryptanalyst can get access to these discards), then he obtains

$$(c')^D \equiv c^D x \equiv m x \pmod{n},$$

and can recover m . Thus the cryptanalyst can decipher any single ciphertext at the cost of using A 's decryption mechanism once.

In this note we present a slightly different attack. As in the Davida-style attacks, it requires that user A decrypt a certain number of chosen ciphertexts and make the decrypted versions available to the cryptanalyst. (We are assuming here for simplicity that user A 's public key (E, n) is used to send private information to him. A similar procedure applies if these keys are used for authentication, in which case a chosen plaintext attack is used.) The differences between our scheme and Davida's is that once the decrypted plaintexts of the chosen ciphertexts are obtained, no further decryptions by A are needed to read individual messages. More precisely, let $L = L(n)$ denote any quantity that satisfies [11]

$$L = \exp((1+o(1)) ((\log n) (\log \log n))^{1/2}) \text{ as } n \rightarrow \infty.$$

Then, if the cryptanalyst succeeds in obtaining decrypted versions of $L^{1/2}$ chosen ciphertexts, he can decipher any specific ciphertext in $L^{1/2}$ bit operations on his own computer. (It is possible to decrease the number of operations required to decrypt individual ciphertexts at the cost of increasing the number of uses of A 's decryption facility, and vice versa.) The importance of this result is that the best currently known algorithms for factoring integers of the same size as n require L bit operations [2,8,9]. (The memory required is $L^{1/2}$ bits for our attack, although it can be a very slow memory, such as a tape. Some factoring algorithms require negligible memory, while others also require $L^{1/2}$.) Therefore our attack on the RSA cryptosystem, although based on very special assumptions, appears to be the most general one that has been proposed so far and is substantially faster than factoring n .

Our basic assumptions are not completely unrealistic. It is easy to imagine situations where decryptions that are not intelligible might not be classified and would be thrown away (either by poorly trained secretaries or by software programs) in a form where they could be intercepted by the cryptanalyst. Also, one can imagine situations in which repairmen servicing either the decryption "black box" or nearby pieces of equipment could obtain the desired decryptions. Another scenario where our attack might apply arises when a whole group of users uses the same decryption key, which is not accessible to them. By using the decryption "black box," any one user can accumulate data required by our attack which would let him break the scheme even after he was denied access to the "black box" if the key was not changed. (The reader could remark that group members can sign all the messages they wish and use them later on, so this method is not necessary. However, not all fraudulent messages that the forgers might wish to use can be predicted beforehand.)

Our attack can also be applied to the well-known "three-pass" system for transmitting information using exponentiation in a finite field [8; pp. 345-346]. In it, user A wishes to send message m to user B, where m is regarded as belonging to some fixed and known finite field $GF(q)$. User A selects an integer a such that $(a, q-1) = 1$ and transmits m^a to B. User B then selects an integer b with $(b, q-1) = 1$ and sends m^{ab} to A. Next, A computes a' such that $aa' \equiv 1 \pmod{q-1}$, and sends $m^{aba'} = m^b$ to B, who now obtains m from $m = m^{bb'}$, where $bb' \equiv 1 \pmod{q-1}$. Should user B always use the same integer b , our attack could again be applied. In it the cryptanalyst would send $L^{1/2}$ messages u to B, (where $L = \exp((1+o(1))((\log q)(\log \log q))^{1/2})$ this time), would receive u^b for each one of them, and would then be able to decipher any messages that might be sent to B using this protocol in $L^{1/2}$ bit operations on his own computer. The same kind of attack applies if user A always uses the same integer a . (This kind of attack could also be applied to the basic Diffie-Hellman key distribution scheme, but in that context is less realistic.)

The basic lesson of our attack is that one has to be very careful in using the RSA cryptosystem and discrete exponentiation schemes to keep them secure. If the attacks that are outlined above have to be guarded against, moduli somewhat larger than those currently being recommended are likely to be required. However, as we note at the end of the next section, in practical situations the necessary increase in the modulus size is likely to be quite small.

2. The attack

Our attack is a modification of an algorithm used for computing discrete logarithms in fields $GF(p)$ for p a prime [2]. Many of the number theoretic estimates that we utilize can be found there and in [11].

Let $\alpha > 0$ be fixed, and let $k = \lfloor n^{1/2} \rfloor$. In the first stage we utilize user \mathcal{A} 's decrypting facility to obtain $x^D \pmod{n}$ for all $x \in S = S_1 \cup S_2$, where

$$S_1 = \{p: p \leq L^\alpha, p \text{ a prime}\}, \quad (2.1)$$

$$S_2 = \{k+1, k+2, \dots, k + \lfloor L^\alpha \rfloor\}.$$

In order to avoid detection by any simple screening program, we choose a random y_x for each x and obtain $(x y_x^E)^D \equiv x^D y_x \pmod{n}$ from the decrypting algorithm, which then allows us to obtain $x^D \pmod{n}$.

Once we have obtained $x^D \pmod{n}$ for all $x \in S$, we can proceed to decrypt individual ciphertexts c . (At this stage we will not need to use the decrypting facility any more.) The basic idea is to find a representation

$$c \equiv y^E \prod_{x \in S} x^{a_x} \pmod{n} \quad (2.2)$$

for some integers a_x and y , since then

$$c^D \equiv y \prod_{x \in S} (x^D)^{a_x} \pmod{n},$$

where y and all the x^D are known to the cryptanalyst as explained above.

To obtain the representation (2.2), we proceed in two stages. In the first stage we find a y and primes $q_i \leq L^{2\alpha}$ such that

$$c \equiv y^E \prod_{i=1}^h q_i \pmod{n}. \quad (2.3)$$

To obtain the representation (2.3), we choose a random y , compute

$$b \equiv c y^{-E} \pmod{n}, \quad 1 \leq b \leq n,$$

and check whether b factors into primes $q \leq L^{2\alpha}$. We expect to test approximately $L^{1/(4\alpha)}$ values of y before a factorization of c of the form (2.2) is found [2,11], and for each y , it takes $L^{o(1)}$ bit operations to test whether such a factorization exists by using Lenstra's elliptic curve factorization algorithm [9]. Therefore this stage is expected to take time $L^{1/(4\alpha) + o(1)} = L^{1/(4\alpha)}$.

Once a factorization of the form (2.3) is obtained, we proceed to the second stage, in which we represent each of the at most $O(\log n) = L^{o(1)}$ primes $q = q_i \leq L^{2\alpha}$ in the form

$$q \equiv \prod_{x \in S} x^{u_x} \pmod{n}, \quad (2.4)$$

where only $O(\log n)$ of the u_x are non-zero (possibly negative). Once such a representation is obtained for each of the q 's, we quickly obtain (2.2).

To see how to represent a prime $q \leq L^{2\alpha}$ in the form (2.4), let

$$m = \lfloor n^{1/2} q^{-1} \rfloor$$

and determine those integers among

$$m+1, m+2, \dots, m + \lfloor L^\beta \rfloor \quad (2.5)$$

that are divisible solely by primes $p \leq L^\alpha$. There will be $L^{\beta-1/(4\alpha)}$ such integers, and finding them will take L^β bit operations if we employ the Lenstra algorithm, for example.

We next consider two cases. If $\alpha \geq 1/2$, we take $\beta = 1/(4\alpha) + \delta$ for any $\delta > 0$. We then have L^δ integers $m+j$, $1 \leq j \leq L^\beta$, all of whose prime factors are $\leq L^\alpha$. For each such integer and any i , $1 \leq i \leq L^{1/(4\alpha)}$, (note that $1/(4\alpha) \leq \alpha$)

$$q(m+j)(k+i) \equiv t \pmod{n}, \quad (2.6)$$

where $t \leq n^{1/2+o(1)}$ and k was defined at the beginning of this section. Therefore, if the t 's factor like random integers of the same size, we will find L^δ t 's that factor into primes $\leq L^\alpha$, and any single one will give us a factorization of the form (2.4), which gives the desired result. Since testing of each t takes $L^{o(1)}$ bit operations, this stage requires $L^{\beta+o(1)}$ bit operations as $n \rightarrow \infty$, and since this holds for all $\delta > 0$, we conclude that for $\alpha \geq 1/2$, this stage can be carried out in $L^{1/(4\alpha)}$ bit operations.

It remains to consider the case $\alpha < 1/2$. Here we take $\beta = 1/(2\alpha) - \alpha + \delta$. We expect to find $L^{\beta-1/(4\alpha)} = L^{1/(4\alpha)-\alpha+\delta}$ values of $m+j$, $1 \leq j \leq L^\beta$, which factor into primes $\leq L^\alpha$, and it takes $L^{\beta+o(1)} = L^\beta$ bit operations to find them. For each one, and for $1 \leq i \leq L^\alpha$, we test whether the t defined by (2.6) is composed of primes $\leq L^\alpha$. We expect to find L^δ of them. Letting $\delta \rightarrow 0$, we discover that this case takes $L^{1/(2\alpha)-\alpha}$ bit operations.

We thus conclude that if the cryptanalyst can obtain decryptions of L^α chosen ciphertexts he will be able to decrypt any individual ciphertext in $L^{1/(4\alpha)}$ bit operations for $\alpha \geq 1/2$, and in $L^{1/(2\alpha)-\alpha}$ bit operations for $0 < \alpha \leq 1/2$. For $\alpha = 1/2$, both stages require $L^{1/2}$ steps. Since the modulus n can be factored in L bit operations, our attack has an asymptotically smaller running time precisely for

$$\frac{-1 + \sqrt{3}}{2} = 0.366... < \alpha < 1.$$

In practice, our $L^{1/2}$ attack does not offer too much of a speed improvement over the L attacks that factor the modulus. The main reason is that for moduli of practical size (500 to 1000 binary bits) $L^{1/2}$ is quite small, and there is no known way to test whether an integer of size about n (or even $n^{1/2}$) is divisible only by primes $\leq L^{1/2}$ that is much more efficient than trial division. The Lenstra algorithm can be avoided by utilizing somewhat more involved, although probably more efficient methods (cf. [2]), but even they probably would not offer too much of a speedup.

References

1. M. Blum, A potential danger with low-exponent modular encryption schemes, to be published.
2. D. Coppersmith, A. M. Odlyzko, and R. Schroepfel, Discrete logarithms in $GF(p)$, *Algorithmica*, to appear.
3. G. Davida, Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem, Tech. Rept. TR-CS-82-2, Dept. of Electrical Engineering and Computer Science, Univ. of Wisconsin, Milwaukee, Wisconsin, Oct. 1982.
4. R. A. DeMillo, G. I. Davida, D. P. Dobkin, M. A. Harrison, and R. J. Lipton, *Applied Cryptology, Cryptographic Protocols, and Computer Security Models*, Proc. Symp. Appl. Math. #29, Am. Math. Soc. 1983.
5. D. E. Denning, Digital signatures with RSA and other public-key cryptosystems, *Comm. ACM* 27 (1984), 388-392.
6. W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory*, IT-22 (1976), 644-654.
7. J. Hastad, On using RSA with low exponent in a public key network, to be published.
8. A. G. Konheim, *Cryptography: A Primer*, Wiley, 1981.
9. H. W. Lenstra, Jr., manuscript in preparation.
10. A. M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, *Proc. Eurocrypt '84*, to appear.
11. C. Pomerance, Analysis and comparison of some integer factoring algorithms, pp. 89-139 in *Computational Methods in Number Theory: Part 1*, H. W. Lenstra, Jr., and R. Tijdeman, eds., Math. Centre Tract 154, Math. Centre Amsterdam, 1982.
12. R. L. Rivest, Remarks on a proposed cryptanalytic attack on the M.I.T. public-key cryptosystem, *Cryptologia* 2 (1978), 62-65.
13. R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Comm. ACM* 21 (1978), 120-126.
14. G. T. Simmons and J. N. Norris, Preliminary comments on the M.I.T. public-key cryptosystem, *Cryptologia* 1 (1977), 406-414.

ON THE DESIGN OF S-BOXES

A. F. Webster and S. E. Tavares
Department of Electrical Engineering
Queen's University
Kingston, Ont.
Canada

The ideas of completeness and the avalanche effect were first introduced by Kam and Davida [1] and Feistel [2], respectively. If a cryptographic transformation is complete, then each ciphertext bit must depend on all of the plaintext bits. Thus, if it were possible to find the simplest Boolean expression for each ciphertext bit in terms of the plaintext bits, each of those expressions would have to contain all of the plaintext bits if the function was complete. Alternatively, if there is at least one pair of n -bit plaintext vectors X and X_i that differ only in bit i , and $f(X)$ and $f(X_i)$ differ at least in bit j for all

$$\{(i,j) \mid 1 \leq i,j \leq n\}$$

then the function f must be complete.

For a given transformation to exhibit the avalanche effect, an average of one half of the output bits should change whenever a single input bit is complemented. In order to determine whether a given $m \times n$ (m input bits and n output bits) function f satisfies this requirement, the 2^m plaintext vectors must be divided into 2^{m-1} pairs, X and X_i , such that X and X_i differ only in bit i . Then the 2^{m-1} exclusive-or sums

$$V_i = f(X) \oplus f(X_i)$$

must be calculated. These exclusive-or sums will be referred to as avalanche vectors, each of which contains n bits, or avalanche variables.

If this procedure is repeated for all i such that $1 \leq i \leq m$, and one half of the avalanche variables are equal to 1 for each i , then the function f has a good avalanche effect. Of course this method can be pursued only if m is fairly small; otherwise, the number of plaintext vectors becomes too large. If that is the case then the best that can be done is to take a random sample of plaintext vectors X , and for each value of i calculate all the avalanche vectors V_i . If approximately one half the resulting avalanche variables are equal to 1 for all values of i , then we can conclude that the function has a good avalanche effect.

THE STRICT AVALANCHE CRITERION AND THE INDEPENDENCE OF AVALANCHE VARIABLES

The concepts of completeness and the avalanche effect can be combined to define a new property which we shall call the strict avalanche criterion. If a cryptographic function is to satisfy the strict avalanche criterion, then each output bit should change with a probability of one half whenever a single input bit is complemented. A more precise definition of the criterion is as follows. Consider X and X_i , two n -bit, binary plaintext vectors, such that X and X_i differ only in bit i , $1 \leq i \leq n$. Let

$$V_i = Y \oplus Y_i$$

where $Y = f(X)$, $Y_i = f(X_i)$ and f is the cryptographic transformation under consideration. If f is to meet the strict avalanche criterion, the probability that each bit in V_i is equal to 1 should be one half over the set of all possible plaintext vectors X and X_i . This should be true for all values of i . Again, unless n is small it would be an immense task to follow this procedure for all possible vector pairs X and X_i .

An alternate method which could be used to ascertain whether a given cryptographic transformation, f , satisfies the strict avalanche criterion would be to construct a dependence matrix. First an n -bit, random plaintext vector X is generated and its corresponding m -bit ciphertext, $Y = f(X)$, is obtained (n and m will be equal if f is an invertible transformation and there is no data expansion). Then the set of n vectors

$$(X_1, X_2, \dots, X_n)$$

is formed such that X and X_j differ only in bit j . The ciphertext vectors

$$(Y_1, Y_2, \dots, Y_n)$$

are then found where $Y_j = f(X_j)$, and they are used to obtain the set of m -bit binary avalanche vectors

$$(V_1, V_2, \dots, V_n)$$

such that $V_j = Y \oplus Y_j$. This procedure is illustrated in Figure 1.

The value of bit i in V_j (either a 1 or a 0) is added to element $a_{i,j}$ in the $m \times n$ dependence matrix A . This procedure is repeated for a large number, r , of randomly generated plaintext vectors X , and each element in A is divided by r . Then each $a_{i,j}$ gives the strength of the relationship between plaintext bit j and ciphertext bit i . A value of 1 indicates that whenever bit j is complemented in the plaintext then the ciphertext bit i will also change its value, while a value of 0 indicates that the ciphertext bit is completely independent of the plaintext bit. If all elements in the matrix have a nonzero value then the cryptographic transformation is complete, and if it is to satisfy the strict avalanche criterion, every element must have a value close to one half. Therefore, completeness is a necessary condition if the strict avalanche criterion is to be met.

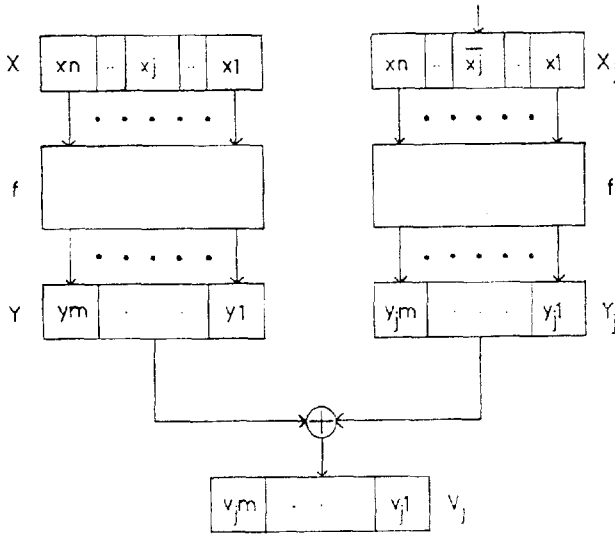


Figure 1. Part of the method for testing to see if a transformation satisfies the strict avalanche criterion: Input bit j is complemented giving V_j . Each bit i , $1 \leq i \leq m$, in V_j is added to element $a_{i,j}$ in the dependence matrix.

A second property which would seem desirable for any cryptographic transformation is that, for a given set of avalanche vectors generated by the complementing of a single plaintext bit, all the avalanche variables should be pairwise independent. In order to measure the degree of independence between a pair of avalanche variables, we can calculate their correlation coefficient. For two variables A and B

$$\rho\{A, B\} = \frac{\text{cov}\{A, B\}}{\sigma\{A\} \sigma\{B\}} \quad [3, \text{p.378}]$$

where

$\rho\{A, B\}$ = correlation coefficient of A and B

$\text{cov}\{A, B\}$ = covariance of A and B

$$= E\{AB\} - E\{A\} \times E\{B\}$$

$$\sigma^2\{A\} = E\{A^2\} - (E\{A\})^2$$

For the case of binary variables, it can be shown that a correlation coefficient of 0 means that the variables are independent. In addition, the variables will always be identical if the correlation coefficient equals 1, and a value of -1 means that they will always be complements of one another [4].

If either the strict avalanche criterion or the avalanche variable independence requirement is not satisfied, then a cryptanalyst can gain some information about the statistical properties of the function, which he could conceivably use to his advantage in an attack on the system.

PERFECT S-BOXES

Now that these two new criteria have been presented, it would seem desirable to discover how to produce cryptographic transformations which satisfy both conditions. One additional condition that will be imposed on such transformations is that they be invertible. This means that there must be a one-to-one correspondence between plaintext and ciphertext vectors.

If there are n input/output bits for a given function, there are $(2^n)!$ possible invertible transformations. This means that there will be approximately 2×10^{13} such functions for a four-bit system. Therefore, the search will be limited to 4×4 (four input/four output bit) substitution boxes (S-boxes).

The initial step is to find all the potentially invertible 4×1 functions that satisfy the strict avalanche criterion, which will be combined four at a time to produce 4×4 substitution boxes. A potentially invertible function returns a value of 1 for one half of the possible input vectors and a value of 0 for the other half. It is a necessary, but not sufficient, condition if the S-boxes formed from the single output bit functions are to be invertible. The 12,870 potentially invertible, 4×1 functions were tested, and it was found that while 12,618 of them were complete, only 1368 satisfied the strict avalanche criterion [4].

These 1368 functions can be divided into 9 equivalence classes or "families". Each family is closed under the following operations:

1. Complementing one or more of the input bits
2. Permuting the input bits
3. Complementing the output bit

Potential invertibility and adherence to the strict avalanche criterion are preserved over these operations.

The simplest procedure to follow in constructing the substitution boxes would be to randomly select potentially invertible, single output bit functions from the list of those that satisfy the strict avalanche criterion. First, these substitution boxes are tested to see if they are invertible. If they satisfy that requirement, they are then examined to see if, when each input bit is complemented, the resulting avalanche variables are pairwise independent. An S-box that displays both of these properties will be referred to as a "perfect" substitution box.

When the method of random selection of single output bit functions was followed, the probability of the resulting 4×4 S-boxes being invertible was only 1.2×10^{-3} , and only one S-box in 7.1×10^5 was perfect [4]. During this search,

the families of single output bit functions which formed perfect S-boxes were noted. In an attempt to reduce the amount of effort required to produce perfect S-boxes, the families from which the 4×1 functions were selected were fixed so that only combinations which had produced perfect S-boxes in the initial search were used. This increased the frequency of occurrence of perfect S-boxes by about a factor of one thousand. Several other approaches were tried which involved relaxing one or both of the strict avalanche criterion and the avalanche variable independence requirement, but none proved to be as good as choosing the single output bit functions from fixed family combinations.

In the process of building these S-boxes, it was discovered that if an S-box is complete, or even perfect, its inverse function may not be complete. This could become important if these inverse functions are used in the decryption process, for it would be desirable for any changes in the ciphertext to affect all bits in the plaintext in a random fashion, especially if there is not much redundancy in the original plaintext. Complete cryptographic transformations with inverses which are complete are described as being two-way complete, and if the inverse is not complete the transformation is said to be only one-way complete.

A COMPLETE S-P NETWORK

Kam and Davida [1] presented a method whereby an entire S-P network could be guaranteed to be complete if all the substitution boxes used in the procedure were complete. This entailed using specially designed bit permutations between the substitution layers. The networks can be of any size as long as

$$n = k^g$$

where

n = the number of input/output bits for the entire network

k = the number of input/output bits for each S-box

g = the number of substitution-permutation stages

Since completeness is a prerequisite if the strict avalanche criterion is to be met, we thought that perhaps by using perfect S-boxes in the system we could come up with a "perfect" system. A complete S-P network with $n = 64$, $k = 4$ and $g = 3$ was implemented. Unfortunately, it turned out that each output bit changed with a probability of only one eighth when a single input bit was complemented. In fact, it can be shown that the probability of an output bit changing will always be 2^{-g} . This was termed avalanche damping. The same test was run with complete S-boxes of the type that Kam and Davida suggested in their paper instead of perfect S-boxes. The mean value of elements in the dependence matrix was slightly higher at 0.19, but their variance was over one hundred times greater than that calculated when the perfect S-boxes were used [4]. In fact, some elements had values as low as 0.01, which represents a significant shortcoming in the system.

This test was repeated for S-P networks with perfect S-boxes and random bit permutations. A plot of the mean and variance of the elements in the dependence matrix is shown in Figure 2. After three rounds, the performance is poorer than that for the complete S-P network, but after about 12 rounds the strict avalanche criterion is satisfied. This result suggests that with the addition of several S-P stages with complete or perfect S-boxes and random bit permutations, a complete S-P network could still be guaranteed to be complete and would probably satisfy the strict avalanche criterion.

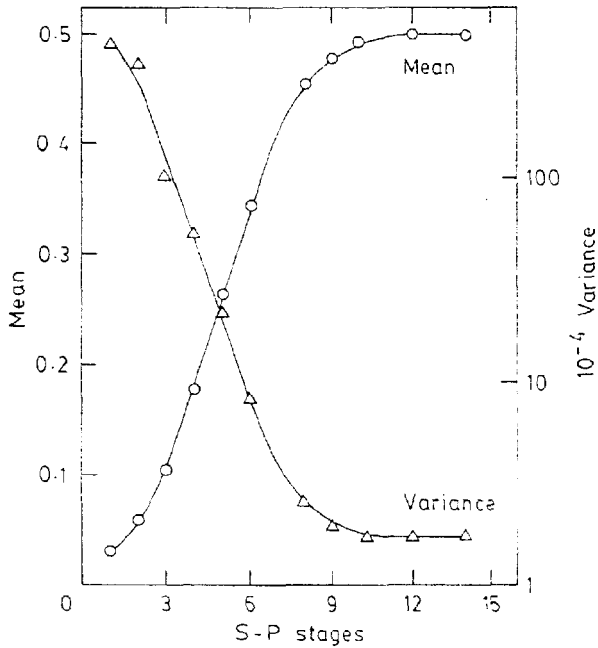


Figure 2. Mean and variance of elements in the dependence matrix for an S-P network vs. number of substitution-permutation stages: All the substitution boxes in this network were perfect, and the bit permutations were generated randomly. It is evident that the strict avalanche criterion is satisfied after approximately 12 S-P stages.

DES

The Data Encryption Standard (DES) has been a federal standard in the United States since 1977. DES employs substitutions and permutations, but the algorithm is much more complex than the one for the complete S-P network [5].

It can be shown that the DES algorithm is invertible [3, p.240]. Since the dependence matrix could, in theory, be different for every key, we cannot state that DES is always a "perfect" system. However, the results shown in Figure

3 for the key (FF . . . FF) indicate that in that case the strict avalanche criterion is satisfied. In addition, in a sample of 30 correlation coefficients picked at random, the highest absolute value found was 4.88×10^{-2} . This suggests that there is very little correlation between avalanche variables. Similar results were obtained using several other key values. Thus, we can conclude that DES is a "perfect" encryption algorithm, at least for the key values that were tested.

Since the S-boxes are the only nonlinear portion of the DES algorithm, their characteristics have a significant effect on the strength of the entire system. The S-boxes are not invertible, but due to the way in which they are employed in the algorithm, this does not pose a problem for decryption. Nor do they satisfy the strict avalanche criterion. For the entire set of 8 S-boxes, the probability that a particular output bit will change when a single input bit is complemented ranges from 0.43 to 0.93.

The correlation coefficients between pairs of avalanche variables for the DES S-boxes were also calculated. While most of them had absolute values of less than 0.5, it was found that when input bit 1 (the least significant bit in the input) was complemented, the correlation coefficients between bits 1 and 2 and between bits 3 and 4 in the output of S_4 were equal to -1. This is equivalent to the discovery made by Hellman et al. [6] that the exclusive-or sums of the output bits, $y_1 \oplus y_2$ and $y_3 \oplus y_4$, of S_4 are complemented whenever input bit x_1 changes its value. It can also be shown that both of these results can be derived from another one of their findings

$$S_4(X \oplus 000001) = (2,1)(3,4) S_4(X) \oplus (x_1, \bar{x}_1, \bar{x}_1, x_1)$$

where (2,1)(3,4) means that the first and second bits as well as the third and fourth bits of the following vector are interchanged.

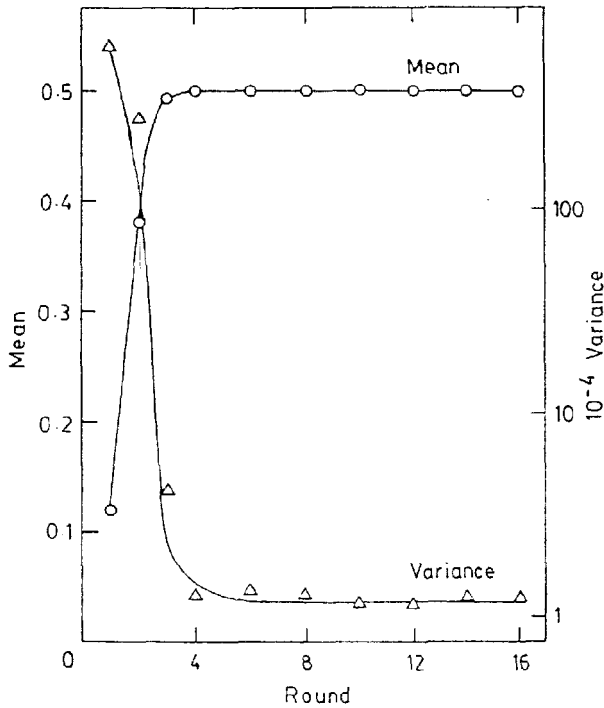


Figure 3. Mean and variance of DES dependence matrix vs. number of encryption rounds: These values of mean and variance are only for elements in one quarter of the dependence matrix, but Meyer [7] shows that these results will propagate through the rest of the matrix within two rounds. The strict avalanche criterion is satisfied after four rounds for this portion of the matrix; therefore, it will take six rounds before the full system will meet the requirement.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support provided by the Natural Sciences and Engineering Research Council.

REFERENCES

- [1] Kam, J.B., and Davida, G.I.: Structured Design of Substitution-Permutation Encryption Networks. IEEE Transactions on Computers, Vol. 28, No. 10, 747 (1979)
- [2] Feistel, H.: Cryptography and Computer Privacy. Scientific American, Vol. 228, No. 5, 15 (1973)
- [3] Konheim, A.G.: Cryptography: a Primer. John Wiley and Sons, New York (1981)
- [4] Webster, A.F.: Plaintext/Ciphertext Bit Dependencies in Cryptographic Algorithms. M.Sc. thesis, Queen's University at Kingston (1985)
- [5] National Bureau of Standards: Data Encryption Standard. FIPS Publication 46, Washington, D.C. (1977)
- [6] Hellman, M.E., Merkle, R., Schroepel, R., Washington, L., Diffie, W., Pohlig, S., and Schweitzer, P.: Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard. SEL 76-042, Stanford University (1976)
- [7] Meyer, C.H.: Ciphertext/Plaintext and Ciphertext/Key Dependence vs Number of Rounds for the Data Encryption Standard. 1978 National Computer Conference, p.1119. AFIPS Press, Montvale, New Jersey (1978)

THE REAL REASON FOR RIVEST'S PHENOMENON

Don Coppersmith
IBM Research
Yorktown Heights, NY 10598

Burt Kaliski, Ronald Rivest and Alan Sherman [Crypto 85] noticed a short cycle in their experiments with weak keys in DES. We explain this in terms of fixed points (messages which are left unchanged by encipherment). We predict similar short cycles using semi-weak keys. We indicate how Rivest *et al's* experimental setup can be used to show that the group of permutations of message space, generated by DES encryptions, is a large group.

Notation: Let $E_K X$ denote the ciphertext resulting from DES-encrypting the cleartext X under the key K . Similarly let $D_K X$ represent decryption. Let 0 denote the key of all 0's, and 1 the key of all 1's. Let the input to DES (the cleartext) be broken into halves M_0, M_1 . On round i , $1 \leq i \leq 16$, we compute some function f of the 48 key bits K_i and the 32 message bits M_i , add this 32-bit quantity to M_{i-1} bitwise, and obtain M_{i+1} . So $M_{i+1} = M_{i-1} + f(K_i, M_i)$, and $M_{i-1} = M_{i+1} + f(K_i, M_i)$. The ciphertext is the pair M_{17}, M_{16} . (Notice the order of indices, which is correct.)

The keys 0 and 1 (and two others) are known to be *weak keys* [Davies, Crypto 82] in the sense that the 48 key bits K_i entering into the computation on round i are the same for each round i : $K_i = K_j$. One consequence of this is that E_0 is an involution: $E_0 X = D_0 X$, so that $E_0^2 X = X$.

A new consequence of being a weak key is that E_0 has 2^{32} fixed points, i.e. messages Y for which $E_0 Y = Y$. Indeed, for some message Y , suppose that $M_8 = M_9$. (There are 2^{32} such values of Y .) Then

$$M_7 = M_9 + f(K_8, M_8) = M_8 + f(K_9, M_9) = M_{10}.$$

Continuing, we find $M_6 = M_{11}, \dots, M_1 = M_{16}, M_0 = M_{17}$, and $Y = (M_0, M_1) = (M_{17}, M_{16}) = E_0 Y$. In fact this is the only way a fixed point can arise for any weak key.

Now pick a random starting message X , and alternately apply E_0 and E_1 . Continue until you return to the starting point X : $(E_1 E_0)^N X = X$. In Rivest *et al's* experiment, N turned out to be around 2^{32} . Indeed, suppose that for some $I < N$, $(E_1 E_0)^I X = Y$ and Y is a fixed point of E_0 . Then the next application of E_0 leaves Y unchanged, so that $E_0 (E_1 E_0)^I X = Y$. On the next application of E_1 , we find

$$(E_1 E_0)^{I+1} X = E_1 E_0 (E_1 E_0)^I X = E_1 Y = D_1 Y = D_1 E_1 E_0 (E_1 E_0)^{I-1} X = E_0 (E_1 E_0)^{I-1} X.$$

Continuing, for $J \leq I$, $(E_1 E_0)^{I+J} X = E_0 (E_1 E_0)^{I-J} X$, and we are just retracing our steps. This is because both E_0 and E_1 are involutions. We continue until, for some $J > I$, $(E_1 E_0)^{I+J} X = Z$ and Z is another fixed point of E_0 . (We could also find fixed points of E_1 .) We again end up retracing our steps, until we return to the starting value X .

The cycle length N is approximately the number of trials we needed to find two fixed points (of either E_0 or E_1 .) Since these fixed points are plentiful (2^{32} out of 2^{64} , or 1 out of 2^{32}), the expected value of N is 2^{32} , in close agreement with Rivest *et al*'s results.

In a similar vein, suppose K is alternating 010101... or 101010... in each key half (a special case of the "semi-weak" keys [Davies]). Let \bar{Z} denote the complement of Z . Then we find that $E_K = D_{\bar{K}}$; there are 2^{32} values Y for which $E_K Y = \bar{Y}$ (namely those for which $M_8 = \bar{M}_0$); and for a random X we expect to find that $E_K^N X = X$ for $N \approx 2^{33}$.

Finally, for different starting values X_i , we expect to find different cycle lengths N_i . Consider the subgroup G of the group of permutations on message space $S_{2^{64}}$ generated by the DES encryptions $E_K, K \in \mathcal{Z}_2^{56}$. Each N_i divides the size of G . Run either of the above experiments several times, finding different values N_i corresponding to $E_1 E_0$ or to E_K for one of the four alternating semiweak keys K . Each experiment takes a few days. Then the least common multiple $lcm(N_1, N_2, \dots, N_i)$ divides the order of the group, and thus provides a lower bound. So the experiments, which were designed to detect a small group size ($|G| < 2^{70}$?) might be used to show a large group size ($|G| > 2^{300}$?).

THE IMPORTANCE OF "GOOD" KEY SCHEDULING SCHEMES (HOW TO MAKE A SECURE DES* SCHEME WITH ≤ 48 BIT KEYS?)

Jean-Jacques Quisquater ^a, Yvo Desmedt ^{b, 1} and Marc Davio ^{a, c}

^a Philips Research Laboratory Brussels,
Avenue Van Becelaere, 2, B-1170 Brussels, Belgium;

^b Katholieke Universiteit Leuven, Laboratorium ESAT,
Kardinaal Mercierlaan, 94, B-3030 Heverlee, Belgium;

^c Université Catholique de Louvain, Bâtiment Maxwell,
Place du Levant, 3, B-1348 Louvain-la-Neuve, Belgium.

Abstract

In DES the key scheduling scheme uses mainly shift registers. By modifying this key scheduling, conventional cryptosystems can be designed which are, e.g., strong against exhaustive key search attacks (without increasing the key size), or have public key like properties. Other effects obtainable by modifying the key scheduling and their importance are discussed.

1. Introduction

In this paper we come up with several ideas which are in contradiction with the common points of view in cryptography. So in the first idea (see Section 2) we will propose to *reduce the key size of a cryptosystem to increase its security against exhaustive key search machines*. This idea sounds crazy, but can be realized for some cryptographic encryption algorithms (e.g. DES) if some very small modifications are used. In the second idea we will come up with a conventional cryptosystem which has public key like properties (see Section 3). In Section 4, we will give examples of *conventional* cryptosystems for which outsiders can prove the *existence* of a trapdoor in the scheme but they cannot use this information to find the trapdoor.

All previous ideas are realized by using new key scheduling schemes.

2. Enforcing cryptosystems against a key exhaustive search

DES [3] was criticized because the length of the key is only 56 bits. Several exhaustive key search machines were presented to break several modes of DES [5], [6], [7], [8], and [9] Diffie and Hellman [7] proposed to use a larger key size to avoid exhaustive key search

¹NFWO aangesteld navorser, sponsored by the National Science Foundation of Belgium.

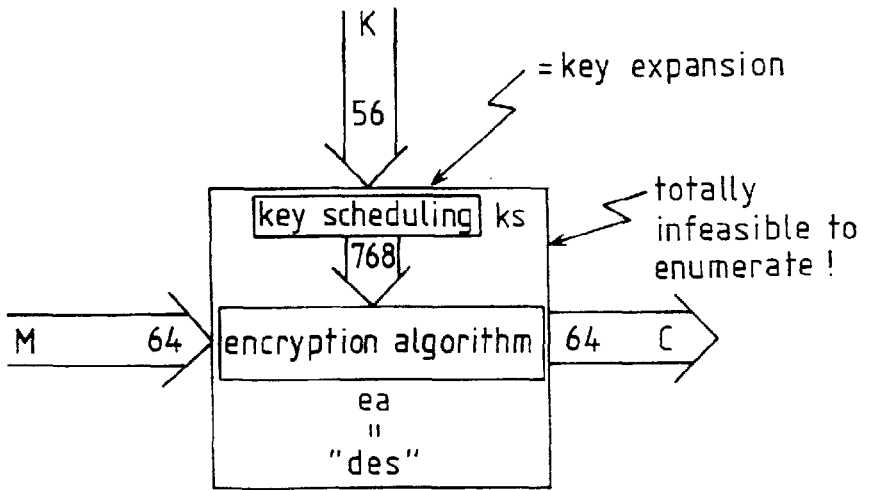


Figure 1: A schematic overview of DES.

attacks. Yasaki [14] cites Hellman: “But the point we’re making here is that a small key guarantees insecurity”. In this section we will explain that in theory an algorithm which uses a short key, can be made very strong against exhaustive key search machines. To realize this in practice some restraints are to be taken into consideration.

To explain the basic idea let us consider DES as an example. The time needed by an exhaustive key search machine to find the key is in worst cases proportional to:

$$\frac{\max(t_{ks}, t_{ea}) \times (\text{size of the key space})}{(\text{number of used processors})}$$

where, for DES, the size of the key space is 2^{56} . As mentioned in Fig. 1: the time t_{ks} , respectively t_{ea} , is the time for executing the key scheduling, respectively the encryption algorithm without the key scheduling. Remark that for many hardware modules DES we have $t_{ea} \gg t_{ks}$. To avoid exhaustive key search attacks, several ideas can be used. The most known is to increase the key space (i.e. key size). Another is to slow down the algorithm (e.g. using more rounds in DES). This solution however reduces the practical use of the algorithm. The solution we present here, is to increase t_{ks} , such that $t_{ks} \gg t_{ea}$. In Fig. 2, a DES-like algorithm is presented which is 16 times harder to break than DES with an exhaustive key search machine! The used key K is 56 bits long and α is some fixed public known 64 bit pattern. Remark that if the key is held constant this DES-like algorithm has the same encryption and decryption speed as DES! Hereto the so called “subkeys” are stored, and were calculated beforehand. Essential for its security is that the DES algorithms in the new key scheduling of the DES-like algorithm (see Fig. 2) are chained. It is evidently necessary that no trapdoor in DES would allow to shortcut this chaining. From now on we will assume that DES doesn’t have the discussed trapdoor.

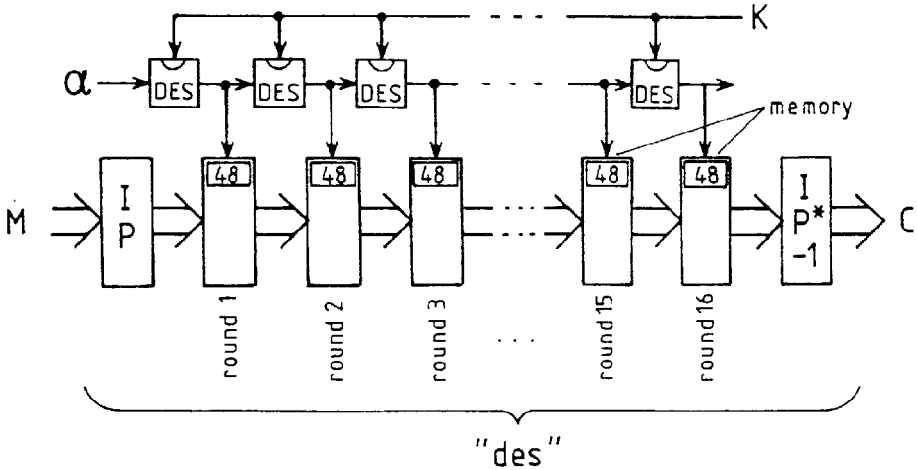


Figure 2: A DES* which is 16 times harder to break with an exhaustive key search machine than DES.

It is now trivial to propose other DES-like algorithms which use a similar key scheduling. *E.g.* one could use 16 times DES for the calculation of the next subkey, instead of one. This would slow down the key scheduling (and the exhaustive key search) with a total factor of 256. Evidently if last discussed scheme is used with a key K of only 48 bits (the other 8 bits can be all zero), the security remains the same as for DES, from the point of view of exhaustive key search attacks. In theory one can increase the key scheduling time as much as we want, however in practice the users modify the key. Then the frequency of the modification of the key determines what an acceptable increase of the execution time of the key scheduling is. However one cannot reduce the key size too far. Indeed if the key is too short, one can precalculate ones and for all the subkeys for all possible keys and store them. In this case exhaustive key search machines use the precalculated subkeys instead of calculating the key scheduling. A similar improved technique is valid for a chosen text attack. Remark that the key scheduling scheme is in fact a key expansion, which is an important concept in modern cryptosystems [11].

Several other schemes can be used as key expansion instead of DES. *E.g.* one can use a modified RSA [13]. The key K of 56 bits is enlarged with zero's to an input for the RSA algorithm of 768 bits, e and n as in RSA are public, however here n is a prime number of 769 bits. The 768 bit output of this RSA scheme is used as the 16 subkeys of 48 bits.

The analysis done on more rounds in DES by the authors [4], is no longer valid for the DES-like algorithm, as a consequence of the harder cryptosystems. In other words the ideas presented here can also be used to make a cryptosystem harder against other attacks than the exhaustive key search.

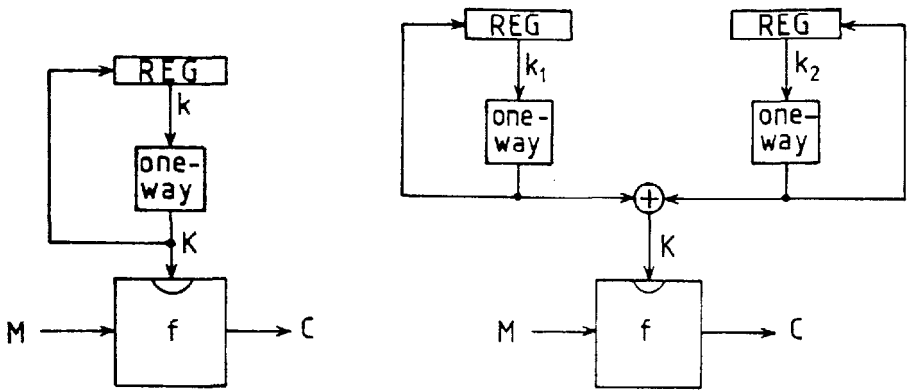


Figure 3: Using a one-way key scheduling scheme in feedback.

3. Using feedback and one-way functions

At the end of the key scheduling scheme in the original DES, the original key reappears in the shift registers C and D . This can be considered as some feedback. Indeed in DES no extra register is necessary to store the used key, if the output of the key scheduling scheme is feedbacked. In the schemes we proposed in this paper, this extra register is necessary, in order to continue to use the same key. This is however not necessary! We can use a feedback at the end of the key scheduling. If sender and receiver remain synchronized and no transmitted bits are lost, the used keys are modified, but both use the correct key. Let us now discuss two cases a little more. In the first one, the key scheduling scheme itself uses a secret master key to calculate the actual "session" key. Remark that the new session key has not to be transmitted! In the second case no extra master key is used. The key scheduling scheme uses only public known information (as in DES). In this case no real advantage seems to be obtained with this feedback. We will now explain that this impression is wrong.

Suppose last discussed feedback key scheduling scheme is used. What happens if the used function ks (see Fig. 3, a) in the feedback is one-way? In that case the conventional cryptosystem acts partially as a public key system. We will explain this by an example. Suppose that a cryptosystem is located in some physical unsafe area. The security of the key is actually tamper free, but this can change at any time (e.g. a bank located in an unstable political regime). One wants to design a cryptosystem, such that if the key of the sender is stolen, the cryptanalyst cannot understand the sent messages. A first possibility is to use a public key system. Indeed the public key (of the receiver), which is used to protect the privacy of the messages, cannot be used to decipher. The second possibility is to use the one-way key scheduling scheme with feedback. Even if the key is stolen, it cannot help to decipher previous messages! Similar examples can easily be found for other areas, in space applications for instance. Applications of the same idea can be used to protect keys in non-tamper free areas, e.g. in chip cards. Indeed chip cards are more

secure than magnetic cards, nevertheless not necessary completely tamper free. Related to this example, a similar scheme was presented by Beker [1].

Several schemes can be used for this one-way function. DES is a good candidate for that. Some caution is necessary because only the so-called leader part of the feedback scheme is useful [10]: The cyclic part of the scheme is not very secure.

Another idea is given at Fig. 3, b. It has the same properties as the one we discussed for the scheme of Fig. 3, a. Additionally the future is protected even if the key K is found by any practical method different from physically stealing the keys k_1 and k_2 .

Intead of one-way functions one can also use hard pseudo-random functions in the sense of Blum and Micali [2].

4. Trapdoors in key scheduling schemes

As we yet discussed in Section 2 trapdoors in the key scheduling are possible. To obtain the improvements, chaining DES must be free of a shortcut solution. In this section more trapdoors in key scheduling schemes will be discussed.

We discussed at the end of Section 2 the use of a modified RSA scheme for key scheduling. We used however there a prime number n , instead of the product of two primes. Suppose however that the user of the cryptosystem verifies if n is indeed prime, and suppose it isn't. He knows for sure that it can be that the one who designed the cryptosystem has deliberately chosen n as a product of primes kept secret by the designer. Using the Chinese remainder theorem this allows the designer to speed up RSA [12] and so the key scheduling and his exhaustive key search machine. Remark that *the user of the cryptosystem can indeed verify the possibility of a trapdoor but cannot use this knowledge!* Evidently if n is large enough, it can be the product of several primes, giving more advantage at the designer. We propose to use the expression "*trapdoor algorithms*" for this kind of algorithms, i.e. for the algorithms where the computation complexity depends on the knowledge of some information.

A trapdoor can also be build in the feedback one-way key scheduling system. Indeed instead of using a one-way function, a trapdoor one-way function can be used. This allow the designer to reverse the feedback and decrypt previous messages, while this is impossible (hard) for outsiders. Remark that the cryptosystem remains a conventional cryptosystem!

Such trapdoors (which are useless for outsiders if they only know the existence and location of the trapdoor) can be used, e.g. to *reduce the misuse* of an authentication system *after* it has been tampered.

5. Conclusions

In contradiction with the common ideas *the key length* is not only *the thing* to protect against exhaustive key search machines. Cryptosystems were proposed acting partially similar as public key systems. Combining ideas of public key and conventional schemes, we proposed trapdoors in conventional systems. The trapdoors are detectable, but useless for outsiders.

A good key scheduling scheme is important. Very hard cryptosystems can be build, starting from simple ones, iterating them and using a hard key expansion (scheduling) scheme.

REFERENCES

- [1] H. Beker and M. Walker, "Key management for secure electronic funds transfer in a retail environment," *Advances in Cryptology*, Proceedings of Crypto '84, Santa Barbara, August 1984 (Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1985), pp. 401 - 410.
- [2] M. Blum and S. Micali "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM J. Comput.*, Vol. 13, No. 4, Nov. 1984, pp. 850 - 864.
- [3] "Data Encryption Standard," *FIPS* (NBS Federal Information Processing Standards Publ.), no. 46, January 1977.
- [4] Y. Desmedt, J.-J. Quisquater and M. Davio, "Dependence of output on input in DES: Small avalanche characteristics," *Advances in Cryptology*, Proceedings of Crypto '84, Santa Barbara, August 1984 (Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1985), pp. 359 - 376.
- [5] Y. Desmedt, "Unconditionally secure authentication schemes and practical and theoretical consequences," presented at Crypto '85, Santa Barbara, August, 1985, to appear in the proceedings: *Advances in Cryptology* (Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1986).
- [6] Y. Desmedt, F. Hoornaert and J.-J. Quisquater, *paper in preparation*.
- [7] W. Diffie and M. E. Hellman, "Exhaustive cryptanalysis of the NBS Data Encryption Standard," *Computer*, vol. 10, no. 6, pp. 74 - 84. June 1977.
- [8] M. E. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Trans. Inform. Theory*, vol. 26, no. 4, pp. 401 - 406, July 1980.
- [9] F. Hoornaert, J. Goubert, and Y. Desmedt, "Efficient hardware implementations of the DES," *Advances in Cryptology*, Proceedings of Crypto '84, Santa Barbara, August 1984 (Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1985), pp. 147 - 173.
- [10] B. S. Kaliski, R. L. Rivest and A. T. Sherman "Is DES a pure cipher? (Results of more cycling experiments on DES)," presented at Crypto '85, Santa Barbara, August, 1985, to appear in the proceedings: *Advances in Cryptology*, (Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1986).
- [11] A. Konheim, "*Cryptography: A Primer*," John Wiley, Toronto, 1981.
- [12] J.-J. Quisquater and C. Couvreur, "Fast decipherment for RSA public-key cryptosystem," *Electronics Letters*, vol. 18, 14th October 1982, pp. 905 - 907.
- [13] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, pp. 294 - 299, April 1978.
- [14] E. K. Yasaki, "Encryption algorithm: key size is the thing," *Datamation*, vol. 22, no. 5, pp. 164 - 166. March 1976.

ACCESS CONTROL AT THE NETHERLANDS POSTAL AND TELECOMMUNICATIONS SERVICES

Willem Haemers
PTT, Dr Neher Laboratories
Leidschendam, The Netherlands

Abstract. The Netherlands Postal and Telecommunications Services (PTT) have developed a system that controls the entrance to their buildings by use of magnetic stripe cards. In this note some cryptographic aspects of the system are explained.

The Netherlands PTT has about 100,000 employees and 2,000 buildings. Many of the employees have access to several buildings. The access control system provides each employee with only one magnetic stripe card, irrespective of the number of buildings the employee has access to. Because of the complexity of the situation an off-line system is preferred. It implies that the access information must be on the magnetic stripe card. The access information consists of the following subjects:

- identity of the employee
- buildings to which the employee has access
- times when access is allowed
- access under special circumstances
- PIN-code
- random information

For reasons of security and organisation it is required that the card distribution center only is able to create cards. This is achieved by encrypting the information by means of a public key system. The secret encryption key, needed to create cards, is then only present at the center, whilst the public decryption key, needed to interpret the cards is present in each building. This kind of public key application can be found in [1] p. 512, and in [3].

Decryption is required to be implemented in PASCAL on a micro computer. A straightforward implementation of RSA takes about one minute. For decoding, this is much too long. Waiting at the entrance should not take more than half a second. One can speed up the decryption of RSA by use of a small exponent. However, Rabin [2] provides a system that in all cases is faster than RSA. The decryption formula for Rabin's system reads

$$(*) \quad (\text{clear text}) = (\text{cipher text})^2 \text{ MOD } (\text{public key}),$$

where, as in RSA, public key is the product of two large primes. Computation of this formula has been realized in about 300 ms (the number size is 480 bits). Encoding still takes about one minute, but this is no problem.

After a card is read at the entrance the card holder can be asked to identify himself by means of a PIN. The PIN is a number chosen by the card owner and has no prescribed length. The information necessary for PIN checking, the PIN-code, is also on the card. If the PIN is typed at the entrance, the PIN-code is computed and compared with the PIN-code on the card. The PIN-code depends on the PIN and the identity of the card owner via a one-way function. The one-way function used is Rabin's decoding formula (*) (only 32 bits of the outcome are taken for the actual PIN-code).

It is impossible to prevent an exhaustive search attack on the PIN by anyone who knows the public key. Therefore the public key is not made public. However, it is straightforward to derive the public key from the plaintext and the ciphertext of about two cards. Therefore knowledge of the full plaintext is prevented by means of the random information on the card. The random information also prevents a chosen

plaintext attack which is known to exist for the used application of Rabin's system.

REFERENCES

- [1] Meyer, C.H. & Matyas, S.M., "Cryptography: A New Dimension in Computer Data Security", John Wiley & Sons Inc., New-York, 1982.
- [2] Rabin, M.O., "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", MIT/LCS/TR-212 (1979).
- [3] Simmons, G.J., "A System for Point-of-Sale or Access, User Authentication and Identification", Proc. Crypto '82, Santa Barbara, pp. 31-37.

Author Index

- Adleman, Leonard M. 3
Berger, Richard 87
Bennett, Charles 468
Blakley, G.R. 180, 282
Boeckee, D.E. 489
Brassard, Gilles 468
Brickell, Ernest F. 28
Chaum, David 18, 192
Chor, Benny 448
Cohen, Gerard D. 458
Coppersmith, Don 14, 104, 535
Crepeau, Claude 73
Davio, M. 537
de Jonge, Wiebren 18
DeLaurentis, John M. 28
Desmedt, Yvo 42, 516, 537
Diffie, Whitfield 108, 340
El Gamal, Taher 396
Estes, Dennis 3
Even, Shimon 58
Evertse, Jan-Hendrik 192
Feigenbaum, Joan 477
Fell, Harriet 340
Galil, Zvi 128
Godlewski, Phillippe 458
Goldreich, Oded 58, 448
Goldwasser, Shafi 448
Gosler, James R. 140
Haber, Stuart 128
Haemers, W. 543
Hastad, Johan 403
Herzberg, A. 158
Kaliski, Burt S. 212
Kannan, Sampath 87
Kochanski, Martin 350
Kompella, Kireeti 3
Luby, Michael 447
McCurley, Kevin S. 3
Meadows, Catherine 180
Miller, Gary L. 3
Miller, Victor S. 417
Moore, T.E. 227
Nakamura, K. 246
Odlyzko, A.M. 516
Okamoto, E. 246
Peralta, Rene 87
Pinter, S. 158
Purdy, G.B. 180
Quisquater, J.-J. 537
Rackoff, Charles 447
Reif, J.H. 433
Reuppel, Rainer A. 260
Rivest, Ronald L. 212
Robert, Jean-Marc 468
Siegenthaler, T. 273
Shamir, Adi 58, 280
Sherman, Alan T. 212
Simmons, Gustavus J. 33
Stephens, N.M. 409
Tavares, S.E. 227, 523
Tygar, J.D. 433
van Tilburg, J. 489
Varadharajan, V. 369
Webster, A.F. 523
Williams, H.C. 358
Wolfram, Steven 429
Yung, Moti 128

Keyword Index

- Algebraic number fields 387
- Algorithm 355, 366
- Alphabet 284
- Arithmetic 284
- Authentication 42, 132
- Authentic public channel 468
- Avalanche variable independence
526, 527
- Basis reduction algorithm 105
- Bhattacharyya-distance 492
- Birthday phenomenon 14
- Bit security 448
- Blockciphers 192
- Boolean polynomials 280
- Buffers 240
- Cantor Set 429
- Cellular automaton 429
- Chosen signature 19
- Chosen text attack 516
- Class number 401
- Closed cipher 214
- Completeness 523
- Computing with encrypted data 477
- Confusion 283
- Conventional cryptosystem 160, 340
- Correlation-Immunity 262, 274
- Cryptanalysis 29, 32, 248
- Cryptographic protocols 58
- Cryptosystem 283
- Cycle detection algorithm 214
- Cycle structure 536
- Cyclotomic field 396, 397
- Cyclotomic polynomial 397, 399
- Data transfer 109
- DES 14, 50, 192, 213, 293, 531, 535,
537
- Digital credit card 461
- Digital signatures 32
- Diffusion 283
- Disclosure rate 248
- Discrete elliptic logarithms 421
- Discrete logarithms 105, 396, 518
- Discrete message 189
- Discriminant 419
- Divide and conquer attack 274
- Elliptic curve 410, 418
- Encryptable problem 477
- End-to-end security 108
- Error probability 489
- Exponentiation 104
- Extended RSA system 382
- F-divergence 492
- Factoring 89, 363, 373, 448
- Factorization 409
- Factorization trapdoor 392
- FAP4 350
- Finger print 180
- Finite field 396, 397, 400
- Finite permutation group 214
- Finite state machine 262, 265, 273
- Flow control 109
- Forger 18
- Forgiving message 180
- Group 284
- Hard bits 129
- Hardware security devices 142
- Ideal classes 399, 401
- Immutable codes 459
- Imperfect private channel 468
- Incremental locked codes 460
- Interdependencies of key bits 193
- J-ring 342
- Jacobi symbol 359
- Key
 - exchange 421
 - lifetime 248
 - distribution schemes 247
 - scheduling scheme 537
- Key-equivocation 489
- Known plaintext attack 192
- Lattice algorithms 405

- Layer interaction 230
- Layer transparency 230
- Layered cryptosystem 228
- Lenstra algorithm 414, 519
- Linear complexity 270
- Linear congruential generators 439
- Link security 108
- Logarithmic height 419
- Logarithms 105
- Matrix 286
- Matrix based RSA system 384
- Matrix ring 377
- Meet-in-the-middle attack 14, 192
- Mental poker 104
- Message authentication 14, 15
- Multi-Party protocols 449
- Multiple encryption 213
- Multivariate polynomial 340
- Nilpotent ring 341
- NP 430
- NP-complete problem 482
- $NP \cap Co-NP$ 488
- One time pad (Vernam) 43
- p-1 method 413
- P-Isomorphism 483
- Parallel computation 435
- Partial factorization 449
- Partial information 448
- Pe-security distance 502
- PIN 543
- Pirate 182
- Polynomial rings 383
- Predicate reducibility 452
- Probabilistic Turing Machine 88
- Protocol
 - coin flipping 88
 - cryptographic 90, 128
 - distribution 165
 - ping-pong 60
 - poker 73
 - replacement 166
 - transmission control 108
- Pseudo random function 447
- Pseudo random permutation 447
- Public key authentication system 28
- Public key cryptosystems
 - 59, 129, 160, 340, 358, 369, 403
- Public key signature scheme 28
- Pure cipher 215, 495
- Quadratic congruences 3
- Ramp scheme 285
- Random number generation 434
- Redundancy 19
- Reliability 109
- Replacement 282
- RNC 435
- RSA chip 350
- RSA cryptosystem
 - 18, 51, 59, 358, 373, 403, 516, 543
- Running-key generator 260, 268
- S-Boxes 280, 527, 529
- Sequences of linear factors 192
- Signatures 3, 18, 144
- Smooth numbers 413
- Software analysis denial 146
- Software protection 140, 158
- S-P Networks 529, 530
- Statistical tests 433, 437
- Stream cipher 49, 260, 273, 429
- Strict avalanche criterion 524, 525
- Subliminal channel 32
- Summation generator 268
- Symmetric encryption scheme 130
- Symmetric group 286
- Symmetrical cryptosystem 232
- Tampering 469
- Technology denial concepts 147
- Three-layered algorithm 238
- Three-way handshake 112
- Transaction system 167
- Transmission errors 469

Transposition cipher 286

Trapdoor

 algorithm 541

 permutation 129

 ring 374

Unconditional security 42, 53

Universal algebra 290

Vector space 284

Weak keys 535

"Write-once" memories 458